DYNAMIC REMOTE DATA AUDITING FOR SECURING BIG DATA STORAGE IN CLOUD COMPUTING

MEHDI SOOKHAK

FACULTY OF COMPUTER SCIENCE AND
INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2015

DYNAMIC REMOTE DATA AUDITING FOR SECURING BIG DATA STORAGE IN CLOUD COMPUTING

MEHDI SOOKHAK

THESIS SUBMITTED IN FULFILMENT
OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE AND
INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2015

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: (I.C./Passport No.:)

Registration/Matrix No.:

Name of Degree:

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

Field of Study:

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date

Subscribed and solemnly declared before,

Witness's Signature Date

Name:
Designation:

ABSTRACT

Nowadays, organizations produce a huge amount of sensitive data, such as personal information, financial data, and electronic health records. Consequently, the amount of digital data produced has increased correspondingly and often overwhelmed the data storage capacity of many organizations. The management of such a large amount of data in local storage system is difficult and incurs high expenses because of high-capacity storage systems needed and the expert personnel to manage them. Although the cost of storage hardware has tremendously decreased in recent years, about 75% of the total ownership cost is still assigned to manage data storage. It is not surprising, therefore, that cloud computing is now embraced as a key technology to provide a convenient, on-demand network access to a shared pool of configurable computing resources, and demands minimum service provider interaction or management effort. Organizations now have an option to outsource their data to cloud storage to decrease the burden on local data storage and also to reduce maintenance cost. Although the cloud offers tangible benefits to data owners, outsourcing data to a remote server and delegating management of data to an untrusted cloud service provider, can lead to loss of physical control over the data. To the clients, the cloud is inherently neither secure nor reliable and this poses new challenges to the confidentiality, integrity, and availability of data in cloud computing. Without a local copy of the data, traditional integrity verification techniques such as hash functions and signatures are inapplicable in the cloud storage. Also, it is impossible to download a large-size file from the cloud storage. The situation is made worse when users access data using their mobile devices. In this context, a more efficient technique is required to remotely verify the integrity of the outsourced data in the cloud. In this research, a new remote data auditing method is proposed for securing data storage in cloud computing based on an algebraic signature. This signature allows the auditor to check data possession in

cloud storage, and this incurs fewer computational overheads on the auditor and server in comparison to Homomorphic cryptosystem. Moreover, a new data structure – Divide and Conquer Table (D&CT) – is designed to efficiently update the outsourced data dynamically by performing insert, append, delete, and modify operations by the data owner. Furthermore, the proposed method is implemented in the real environment to prove the security, justify the performance of our method, and compare with the most familiar and the stat-of-the-art data auditing methods on the basis of computation and communication cost. It is found that by employing the proposed RDA method the computational and communication costs of data integrity is reduced. D&CT data structure reduces the computation cost of data update for normal and large-scale files markedly. Hence, the proposed RDA provides an efficient and secure solution for mobile cloud computing.

ABSTRAk

Dalam dunia hari ini, organisasi menghasilkan sejumlah besar data sensitif seperti maklumat peribadi, data kewangan, dan rekod kesihatan elektronik. Kepesatan generasi data digital telah mendahului keupayaan penyimpanan organisasi itu sendiri. Pengurusan sejumlah besar data adalah sukar untuk disimpan sendiri dan telah meningkatkan perbelanjaan yang tinggi terhadap organisasi kerana memerlukan sistem penyimpanan yang berkapasiti tinggi serta kakitangan yang terlatih. Walaupun kos perkakasan penyimpanan semakin berkurangan, pengurusan simpanan besar itu adalah lebih kompleks yang mana merupakan kira-kira 75% daripada jumlah kos pemilikan. Akibatnya, kebanyakan organisasi cuba untuk menggunakan data penyumberan luar kerana ia mengurangkan beban penyimpanan data tempatan dan mengurangkan kos penyelenggaraan. Apabila pengguna awan menyumber luar data di pelayan capaian jauh, kawalan fizikal ke atas data itu dilepaskan dan pengurusan data tersebut diserahkan kepada Penyedia Perkhidmatan Awan yang tidak boleh dipercayai. Oleh kerana Perkomputeran awan itu pada asasnya tidak selamat serta tidak boleh dipercayai pada pandangan pelanggan, ia menimbulkan cabaran baru kepada integriti data dalam perkomputeran awan. Walau bagaimanapun, teknik pengesahan integriti secara tradisional, seperti fungsi hash dan tandatangan adalah tidak berkenaan dalam perkomputeran awan kerana ketidakupayaan untuk menyimpan data salinan tempatan. Sebaliknya, muat turun fail saiz besar adalah tidak praktikal. Keadaan yang dinyatakan di atas menjadi lebih teruk apabila pengguna mengakses data dengan menggunakan peranti mudah alih sumber terhad. Oleh itu, teknik yang cekap diperlukan dari jauh bagi mengesahkan integriti data yang disumber luar dalam perkomputeran awan. Dalam kajian ini, kami mencadangkan satu skim pengauditan data jauh baru untuk menjamin penyimpanan data dalam perkomputeran awan. Kami juga merekabentuk struktur data baru yang membolehkan pemilik data dengan cekap mengemaskini

data yang disumber luar secara dinamik dengan melakukan sisipan, menambah, memotong, mengubah suai dan operasi. Selain itu, kami menguji skim ini untuk memastikan keselamatan, kewajaran pelaksanaan kaedah ini, dan membandingkan dengan kaedah pengauditan data stat-of-the-art berdasarkan asas pengiraan dan kos komunikasi.

ACKNOWLEDGEMENTS

All praise and glory is due to God for blessing, leading, and strengthening me every single moment of my life. God is always here for me whenever I needed help and guidance.

My deepest gratitude to my supervisor, Prof. Abdullah Gani whose ceaseless support, and encouragement, I will never forget. Prof. Abdullah has guided me throughout the years of my studies. The cornerstone to complete this thesis was my regular meetings with Prof. Abdullah to discuss and review my work several times to be in the absolutely best form we can see. I do appreciate his generosity for making time for my questions on any matter.

Thanks are not enough to be given to my parents, for their ongoing and endless love. There are no words that are valuable to equalize their love for me.

This thesis would not have been possible without the support of my beloved wife, Mahboobeh, who always support me and encouraged me to pursue what I wanted. I would not have been able to achieve any success in my life. She is always my first resort whenever I am tired or frustrated. I enjoyed sharing every moment with her, and I hope I can be a source of her happiness at all times.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ACRONYMS

3DES	Triple-Data Encryption Standard.
ABE	Attribute-Based Encryption.
AES	Advanced Encryption Standard.
BA	Batch Auditing.
CBS	Commitment-Based Sampling.
CC	Cloud Computing.
CDH	Computational Diffie-Hellman.
CM	Cryptography Model.
Compact POR	Compact Proof Of Retrievability.
CRH	Collision-Resistant Hash.
CSP	Cloud Service Provider.
D&CT	Divide and Conquer Table.
Dep	dependability.
DES	Data Encryption Standard.
DO	Data Owner.
DPDP	Dynamic PDP.
DR	Data Recovery.
DRDA	Dynamic Remote Data Auditing.
Dynamic POR	Dynamic Proof Of Retrievability.
E-PDP	Efficient PDP.
EC2	Elastic Cloud Computing.
EPP-PDP	Efficient privacy preserving PDP.
FD-POR	Fair-Dynamic Proof Of Retrievability.
FEC	Forward Error Checking.
HCS	Hash-Compress-and-Sign.
HLA	Homomorphic Linear Authenticator.
HVT	Homomorphic Verifiable Tag.
I-PDP	Improved PDP.
I-POSD	Improved POSD.
IaaS	Infrastructure as Service.
LI	Logical Index.
MHT	Merkle Hash Tree.
ORAM	Oblivious Random Access Machine.
PA	Public Auditing.
PaaS	Platform as Service.
PCAD	Public Cloud Auditing with Deduplication.
PDP	Provable Data Possession.
PDP-based	Provable Data Possession-based.
POR	Proof Of Retrievability.
POR-based	Proof Of Retrievability-based.
POSD	Proof of Storage with Deduplication.
POW	Proof of Ownership.
POW-based	Proof Of Ownership-based.
PP-PDP	privacy preserving PDP.
PPDP	Proxy PDP.
PRE	Proxy re-encryption.
PT	Protocol Type.
Public PDP	Public Provable Data Possession.

Public POR	Public Proof Of Retrievability.
rb23Tree	Range-based 2-3 Tree.
RBAC	Role-based access control.
RD-PDP	Robust Dynamic PDP.
RDA	Remote Data Auditing.
RS	Reed Solomon.
S-PDP	Secure PDP.
S3	Simple Storage Service.
SaaS	Software as Service.
Sec-PDP	Secure PDP.
SN	Scheme Nature.
SRDA	Static Remote Data Auditing.
TPA	Third Party Auditor.
VLGG	Variable Length Constrain Group.
VN	Version Number.

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

This chapter provides an overview of the research work by presenting the statement of problem, objectives of the research, and describing the methodology used for the research. Section 1.2 justifies the motivations for the research and explains the significant of the proposed work. Section 1.3 presents the problem statement by emphasizing on the security issues in the data outsourcing frameworks. Section 1.4 states the research objectives, and Section 1.5 gives an overview on the adopted methodology for the proposed research. Finally, the layout of the thesis is presented in Section 1.6.

1.1 Background

Cloud Computing is a new model of computing over a shared pool of computing resources such as network bandwidth, servers, storage, processing power, applications, and services (Armbrust et al., 2010; Mell & Grance, 2011). Today, this new paradigm has become popular and is receiving a lot of attention from researchers in the academic and industrial communities. A recent survey indicates that more than 79% of organizations attempt to utilize data outsourcing because it relieves the burden of local data storage and reduce the maintenance cost (Buyya, Yeo, Venugopal, Broberg, & Brandic, 2009; Khan, Kiah, Khan, Madani, & ur Rehman Khan, 2013). Moreover, the users are able to access information from anywhere and at anytime, and on any device. (C. Wang, Wang, Ren, & Lou, 2009; Xie, Wang, Yin, & Meng, 2007).

Although cloud computing offers several advantages for users (such as on-demand, affordable, elasticity, ubiquitous resource access and measured service), there are some security concerns that prevent a full adoption of this new technology, such as data in-

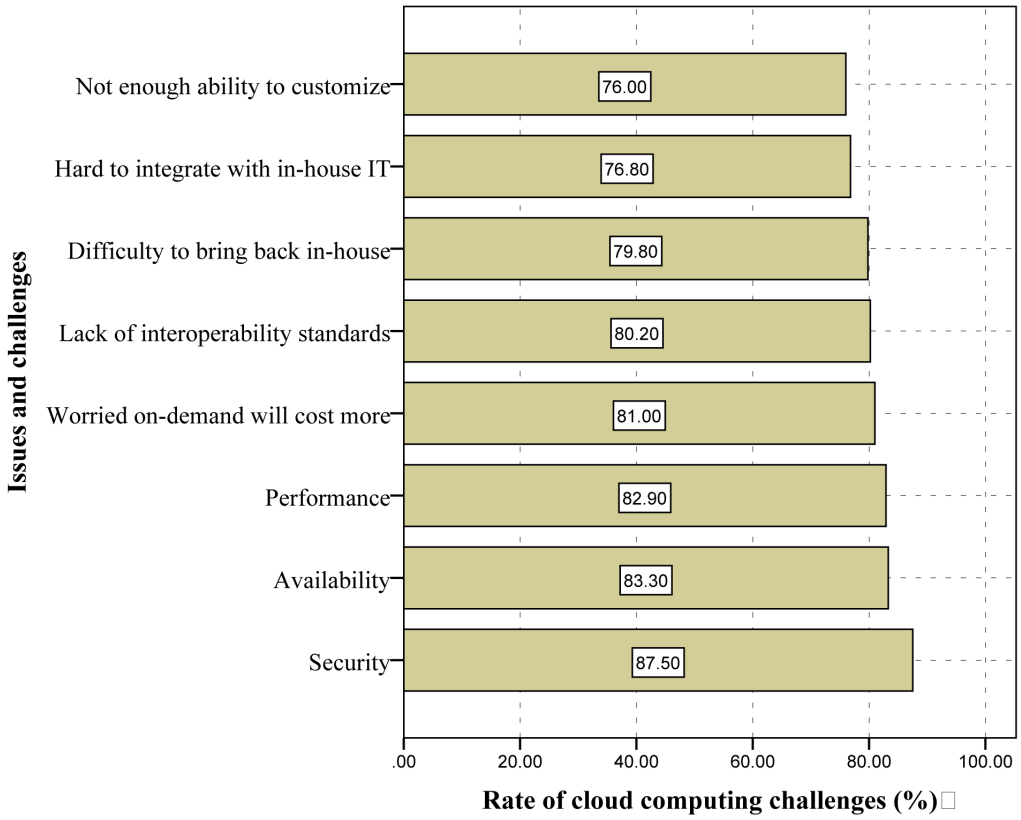


Figure 1.1: Cloud Computing Challenges (Gens, 2010)

tegrity, confidentiality, and availability (Chow et al., 2009; Zhibin & Dijiang, 2012). This is because the major challenge in cloud computing is related to security of data and infrastructure.

International Data Corporation (IDC) recently conducted a survey about the challenges and issues of cloud computing (Christiansen, Kolodgy, Hudson, Pintal, & IDC, 2010). The survey indicates, on the basis of user concerns, the cloud computing challenges are as follows: enough ability to customize, capability to integrate with in-house IT, interoperability standards, cost, performance, availability, and security. Among these cloud computing challenges, security represents 87.5% of users' cloud fears that justify the important of security in terms of users' perspective. Figure 1.1 shows the results of the IDC survey indicates the cloud computing challenges, and the different percentages of users' cloud fears (Gens, 2010).

After outsourcing the data to the remote clouds, the cloud users need to be ensured

that the data remains intact. It is because the physical control is taken away over the data, and the management of the data is delegated to the cloud service provider as an untrusted party (Wei et al., 2013). Even though the cloud resources are more reliable and have more powerful infrastructure than personal computer, the data in the cloud is still vulnerable to many inside and outside threats. Such threats might compromise the confidentiality, integrity, and availability of data (Zhifeng & Yang, 2013). For example, the Byzantine failure is a type of internal attack that may be made by hardware errors or the cloud maintenance personnel's misbehaviors, while external attacks could be ranging from natural disasters, like fire and earthquake, to adversaries' malicious hacking. Moreover, if the adversaries gain the control of the cloud server, they have the capability to launch the forge attack or the replay attack which aims to break the linear independence among encoded data, by replacing the data stored in the corrupted cloud server with old encoded data. As a result, the integrity of cloud users' data stored on the server is at risk. However, traditional integrity verification techniques, such as hash functions and signatures are inapplicable in cloud computing because the data owner no longer physical possess the stored data (Ateniese, Pietro, Mancini, & Tsudik, 2008). On the other hand, downloading of possibly a large-size file is impractical. The aforementioned situation worsens when users are accessing data using mobile devices.

As a conclusion, the cloud users require a reliable audit service to remotely audit the integrity of the outsourced data within the cloud (H. Chen & Lee, 2013). However, due to the externalized aspect of outsourcing data in cloud and mobile cloud computing, it is highly challenging to protect the integrity and privacy of data, secure access to applications and information, and support data and service availability. This research focuses on the remote data auditing methods to securely and efficiently verify the integrity of the data over a cloud managed by the untrustworthy provider without having to retrieve the data.

1.2 Motivation

In today's world, the organizations produce a huge volume of sensitive data, such as personal information, financial data, and electronic health records. The rate of producing digital data consecutively is increasing and overtaking the storage ability of many organizations. The management of such a large amount of data locally is difficult and incurs high expenses on the organizations because of the requirement of high-capacity storage systems and expert personnel. Despite the fact that the cost of storage hardware is decreasing, the management of such huge storage is more complex and requires approximately 75% of the total ownership cost (Singh & Liu, 2008; Y. Chen & Sion, 2011). As a result, most of the organizations attempt to utilize data outsourcing to overcome such difficulties (Buyya et al., 2009).

However, since that data owners delegate the control over the data to an untrusted cloud service provider (CSP), it raises new challenges to the integrity of data in cloud computing systems (Cong, Kui, Wenjing, & Jin, 2010; Wei et al., 2013). A wide range of internal and external security challenges has the capability to endanger the cloud infrastructures. Such threats might compromise the confidentiality, integrity, and availability of data (Khan, Mat Kiah, Khan, & Madani, 2013; Q. Wang, Wang, Li, Ren, & Lou, 2009; Zhifeng & Yang, 2013). In recent times, various companies reported data corruption in servers with major cloud infrastructure providers, and many events of cloud service outages, such as, Amazon Simple Storage Service breakdown (Gohring, 2008), Gmail mass deletion (Arrington, 2006), sidekick cloud disaster (Cellan-Jones, 2009), and Amazon EC2 service outage (Miller, 2010; Whittaker, 2012). Moreover, the Privacy Rights Clearinghouse (PRC) reports more than 535 data breaches during 2013, namely: breaching of the cloud-based email service providers in Epsilon (Storm, 2011), compromising Sony PlayStation Network (L. B. Baker & Finkle, 2011), Sony Online Entertainment,

and Sony Pictures, stealing customers' information on EMC's RSA, and stealing of 3.3 million patients' medical data of Sutter Physicians Services (Schwartz, 2012).

Data breach poses crucial threats to the outsourced data in cloud storage wherein an individual's name plus a medical record and/or a financial record or debit card is potentially put at risk (W. Baker et al., 2011). Data breach usually occurs in different enterprises for a reason of malicious or criminal attack, system glitch, or human error. Ponemon Institute under the sponsorship of IBM has issued its annual report about cost of the data breach from more than 250 organizations of the eleven countries participated in 2014, such as Australia, Brazil, France, Germany, India, Italy, Japan, United Kingdom, United States and, for the first time, Saudi Arabia and the United Arab Emirates. The research shows that the U.S. experienced the highest total average cost of the data breach at more than $5.85 million, followed by Germany at $4.74 million. In sharp contrast, Brazilian and Indian companies experienced the lowest total average cost at $1.61 million and $1.37 million, respectively ("Cost of Data Breach Study: Global Analysis," 2014). Figure 1.2 illustrates the average organizational cost of data breach varies by country in 2013 and 2014.

As a conclusion, the cloud is inherently neither secure nor reliable from the view point of the clients. Recently, a number of remote data auditing framework are proposed to verify the integrity of the outsourced data in the cloud storage (Ateniese et al., 2011; Erway, Küpçü, Papamanthou, & Tamassia, 2009; Yang & Jia, 2013). Although the traditional remote data auditing methods preserve the integrity of data, they suffer from the following issues: (1) They incur high computation cost on the data owner or the authorized auditor; (2) Most of them are unable to support dynamic data updated caused for imposing additional computation and communication cost on the data owner; and (3) By increasing the size of file, the existing data auditing methods incur huge computation cost on the auditor due to the applied data structure on them.

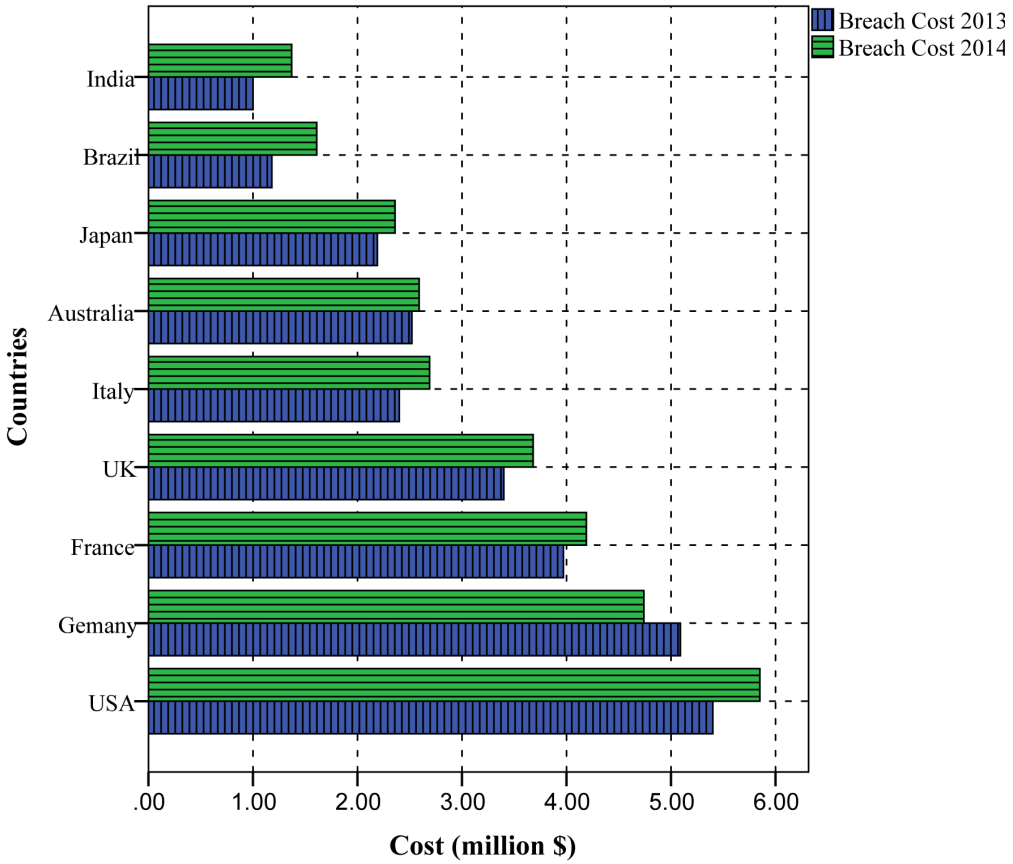


Figure 1.2: The average total organizational cost of data breach over two years ("Cost of Data Breach Study: Global Analysis," 2014)

This thesis addresses these security issues by proposing a secure and reliable remote data auditing method to check the integrity of the outsourced data based on algebraic signature properties. We also present the design of a new data structure that efficiently supports dynamic data operations such as append, insert, modify and delete. Moreover, this data structure empowers our method to be applicable for large-scale data storage with minimum computation cost.

1.3 Statement of Problem

When cloud users store data in remote servers and delete a local copy of data, the physical control over the data is under risk due to the management of data is delegated to a semi-trusted CSP. As a result, the data owners (who outsourced data to the cloud storage) need to use the remote data auditing techniques to securely prove the intactness

of the data resides in cloud storage administrated by semi-trustworthy service provider without downloading the whole data. The remote data auditing methods requires frequent auditing which involves many processes and frequency includes transmission processes. Consequently, the remote data auditing method incurs additional cost due to processing time and transmission processes on the data owner, which is a significant burden for many data owners especially when they use the mobile devices with restricted computing resources (CPU).

Nowadays, many of organizations that produce a huge volume of large-scale data, prefer to archive the data in the cloud storage to reduce the maintenance cost. These organizations must have capability to perform the update operations (delete, insert and modify) on the outsourced data rarely. The RDA methods also have to dynamically support data update operation without downloading the whole data blocks or modifying the rest of blocks. To achieve this goal, the existing RDA methods use different type of data structure (e.g. binary tree) to prove that the outsourced data remains intact. However, the applied data structures in the traditional RDA methods are unable to effectively support dynamic data update operations for large-scale data. In other words, when the size of the outsource file is large, to update a small number of data blocks, the data owner requires re-balancing huge number of data blocks in the data structure. Therefore, the traditional remote data auditing methods incur noticeable processing time on the auditor for rearranging such huge number of blocks.

In contrast to store archival data in the cloud that require rare update operation, there exist some large-scales data with specific application (e.g. business transactions and online social networks), which are intrinsically liable to frequent data update from users (Liu et al., 2014). However, supporting such frequent data update operations using traditional remote data auditing methods result in considerable amount of computation cost for the auditor. This is because the auditor requires rearranging the large number

of data blocks within data structure for several times. As a result, designing a new data structure to support frequent dynamic update for large scale data is imperative.

1.4 Statement of Objectives

We aim to propose an effective remote data auditing method for cloud computing to minimize computational and communication cost of frequent data auditing. The objectives of the research are listed as follows.

- To study the state-of-the-art methods for auditing data in single and distributed cloud server and investigate the additional computation and communication cost in current remote data auditing methods.
- To propose an efficient remote data auditing method for data storage in cloud computing which minimizes computation and communication cost on client and cloud server.
- To design an effective dynamic remote data auditing scheme for various volume of data (e.g. normal and large-scale) by introducing a new data structure.
- To evaluate the proposed method by using mathematical modeling and testing in the emulation environment and validate the performance by benchmarking and comparing the result of different experimental scenarios.

1.5 Proposed Methodology

We describe a literature review of the remote data auditing techniques in a single and distributed cloud server. Remote Data Auditing refers to a sampling of the collected data in the cloud and evaluating the data with various criteria, such as validity, accuracy, and integrity as a way to verify the reliability of the storage provider. The thematic taxonomy of the single and distributed storage auditing schemes are also presented. We identify

the issues and problems of the existing remote data auditing frameworks, which directly affects on the clients and cloud server and impedes the optimization goals of auditing schemes for cloud and mobile cloud computing.

The research problem is investigated by using quantitative analysis on the developed private cloud for checking computation, communication and storage overhead of the static remote data auditing. Therefore, the static remote data auditing method (Ateniese et al., 2007, 2011) is implemented by using the C language and the effects of dynamic data update is evaluated on it. To investigate the effect of data update operations on the dynamic remote data update methods (Q. A. Wang, Wang, Ren, Lou, & Li, 2011; Yang & Jia, 2013), their data structures are implemented by using C++ and Java language. We use different file size to explore the computation time and communication cost of the dynamic data update in different scenarios.

We propose a novel remote data auditing framework for cloud computing to address the issues of existing auditing schemes. We also design a new data structure to efficiently support dynamic data operations, such as insert, append, delete, and modify operations. As a result of implementing this data structure, the proposed model is applicable for auditing large scale data storage dynamically.

The performance of the proposed model is evaluated by using mathematical modeling. We also validate the proposed model by using synthetic workload for mobile devices. We set up our own Eucalyptus private Infrastructure as a Service (IaaS) cloud in order to conduct the experiment. The experimental data are collected by implementing the framework in different scenarios to evaluate the computation and communication overhead on the auditor and cloud server.

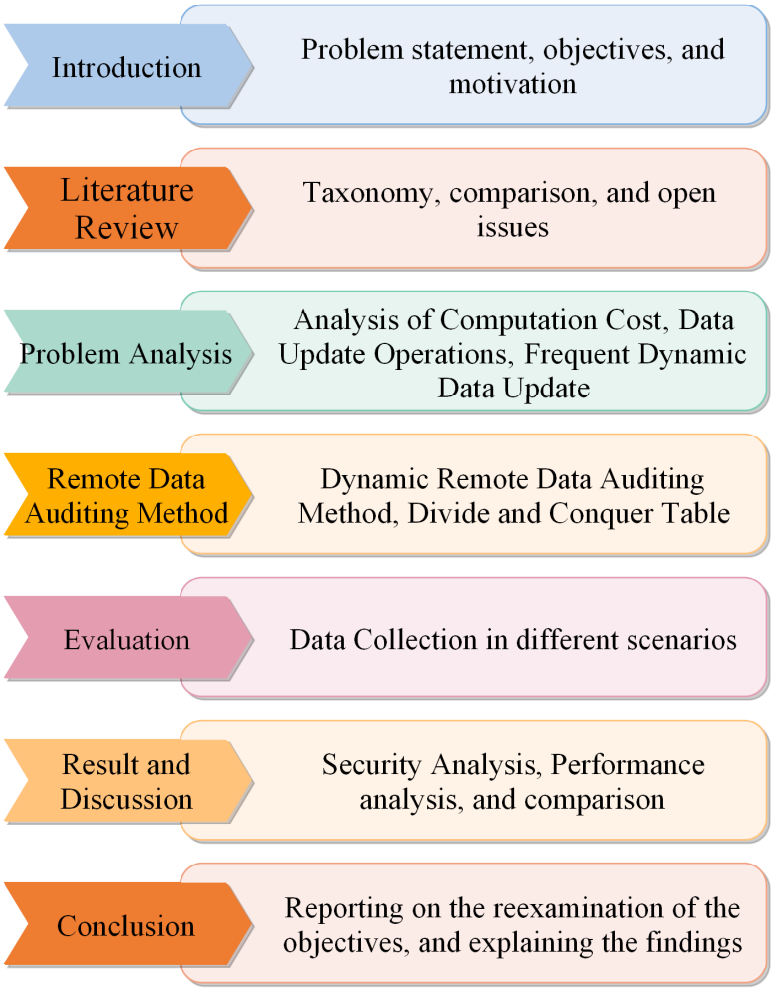


Figure 1.3: Outline of the thesis

1.6 Layout of Thesis

This thesis is composed of seven chapters. Every chapter of this thesis is divided into three parts; Introduction – to state the objective of the chapter; Material – to present the main body of the chapter; Conclusion is a judgment or evaluation of the chapter objective achievement and linkage to the next chapter. Figure 1.3 presents the outline on this study.

Chapter 2: Review of Literature.

This chapter presents a review of the state-of-the-art remote data auditing techniques in single and distributed cloud server domains. The objective of this chapter is to highlight issues and challenges to current RDA protocols in the cloud and the mobile cloud computing. The thematic taxonomy of the data storage auditing is presented based on significant parameters, such as security patterns, objective functions, auditing mode, up-

date mode, and dynamic data structure. The state-of-the-art RDA approaches that have not received much coverage in the literature are also critically analyzed and classified into three groups: provable data possession, proof of retrievability, and proof of ownership to present a taxonomy. It also investigates similarities and differences in such methods and discusses to diagnose significant and outstanding issues for further investigation.

Chapter 3: Problem Analysis.

This chapter explores the computation and communication overhead of the remote data auditing scheme on the client and cloud sides to justify the research problem. We examine the additional overhead of dynamic data update operations by using the emulation environment. The measurement parameters for problem analysis include computation cost, and communication cost of during the verification phase, computation and communication overhead for updating a block in static and dynamic methods; computation cost of frequent data updates for normal file size, and computation cost of updating large-scale files.

Chapter 4: Remote Data Auditing Method.

This chapter presents the method used to achieve the objective of the research. It outlines and streamlines the proposed framework for auditing data storage and elucidates the architecture of the proposed model. A new data structure is also designed that is called divide and conquer table to effectively support dynamic data update operations in block level, such as: modification, insert, delete, and append. This data structure empowers the proposed method to be applicable for auditing large-scale file size.

Chapter 5: Evaluation.

This chapter clarifies techniques that are used to experiment and presents the data collection for evaluating the computation and communication cost of the proposed method in distinctive phases, such as: setup, challenge, response, and verification. It also explains the setup environment, programming tools that are used to implement the pro-

posed method, the measures that are used to check the effectiveness, and the method used to processing the data.

Chapter 6: Result and Discussion.

The aim of result and discussion chapter is to show the advantage of the proposed remote data auditing method, the dynamicity feature for normal and large-scale files, and evaluate the performance of this method. To achieve these goals, the performance and usefulness of the proposed method are presented by analyzing the experimented results. This chapter analyze the security and correctness of the proposed remote data auditing construction. Furthermore, the experiential result of the computation cost and computation of the proposed method in different phases are explained base on distinctive parameters. The significant and validation of the proposed method is also discussed by analyzing and comparing the result in different scenarios.

Chapter 7: Conclusion

This chapter concludes the thesis by reporting on the addressed objectives of the thesis. Moreover, it presents the finding of the research and focusing on the significant of the proposed method. The scope, limitations and directions for future research this research is also explained in this chapter.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter reviews the existing remote data storage auditing schemes to verify the integrity of outsourced data in the single cloud server domain. The objective of this chapter is to highlight issues and challenges to current RDA protocols in the cloud and the mobile cloud computing. It discusses the thematic taxonomy of RDA based on significant parameters such as security requirements, security metrics, security level, auditing mode, and update mode. The state-of-the-art RDA approaches that have not received much coverage in the literature are also critically analyzed and classified into three groups of provable data possession, proof of retrievability, and proof of ownership to present a taxonomy. It also investigates similarities and differences in such methods and discusses open research issues as the future directions in RDA research.

The chapter is structured as follows. Section 2.2 presents the fundamental concepts of cloud computing and mobile cloud computing. Section 2.3 discusses the concept of RDA, and explains the architecture of remote data auditing schemes. Section 2.4 taxonomizes and reviews the state-of-the-art RDA approaches and investigates the critical aspects of the current auditing schemes. Section 2.5 provides a comparison of the current RDA techniques by using the similarities and differences of the significant parameters presented within the taxonomy. Section 2.6 focuses on the issues and challenges in current RDAs. Finally, section 2.7 concludes the paper with future directives.

2.2 Background

This section describes the concepts of cloud computing and fundamental of cryptographic algorithm, such as homomorphic encryption and algebraic signature.

2.2.1 Cloud Computing

The Cloud Computing (CC) has emerged as the latest utility-oriented distributed computing model and has been envisaged as the next generation of Information Technology (IT), with the aim of augmenting the capabilities of the client devices by accessing a pool of leased platforms, infrastructures, and applications without having to actually own them. The cloud service models offer low-cost, on-demand, ubiquitous resource access, rapid elasticity or expansion, and measured services (Höfer & Karagiannis, 2011; Whaiduzzaman, Sookhak, Gani, & Buyya, 2014). The cloud systems have the capability of conveniently adjusting the virtual allocated resources on the basis of the current requirements with a minimal managerial effort and service interruption. Such elastic characteristics reduce the wastage of resources in case of over-provisioning (Aceto, Botta, de Donato, & Pescapè, 2013; Manvi & Krishna Shyam, 2013).

The cloud service models rely on a pay-as-you-go pricing model that charges the clients on the basis of the amount of usage and some service metrics (Zhifeng & Yang, 2013). For example, the Dropbox service can be measured as Gigabyte per year. The cloud computing also has led into appearing a new type of communication and collaboration services by creating online social networks in which scientists are able to construct research communities by sharing data and analysis tools (Barga, Gannon, & Reed, 2011). The virtualization of resources is the core technology of cloud computing to inculcate a vision of infinite resources to the clients (Fernando, Loke, & Rahayu, 2013).

From the perspective of deployment, the cloud computing is classified into four modes, namely: public, private, hybrid, and community clouds, which are details as fol-

lows: (1) Public cloud: In public cloud, service providers deliver application as services over the Internet and allow clients to access computing resources through centralized cloud servers, such as Amazon Web Services, Google App Engine, Microsoft Azure platform, Salesforce, and Aneka (Fox et al., 2009). The Amazon Web Services allow users to store data in Simple Storage Services (S3) (Kristensen, 2007), Google App Engine offers deployment platforms and hosts web applications in Googles data centers (chun Wesley, 2011). The Microsoft Azure platform provides a powerful platform for building, deploying web applications in Microsoft data centers, and Aneka is used as a platform to build applications and deploy them on public or private clouds (Calheiros, Vecchiola, Karunamoorthy, & Buyya, 2012). (2) Private Cloud: The services and infrastructure are exclusively used and managed by a single organization. (3) Community Cloud: The services and infrastructure are shared among a set of organizations with common interests or objectives that are managed either internally or by a trusted third party (Marinos & Briscoe, 2009). (4) Hybrid Cloud: A combination of two or three of the aforementioned clouds with multiple providers (Q. Zhang, Cheng, & Boutaba, 2010; Zissis & Lekkas, 2012).

Cloud service providers offer three types of service models, such as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) (Vaquero, Rodero-Merino, & Morán, 2011). In the SaaS layer, the users can access any kind of software service remotely from the mobile devices. For example, Microsoft Live Mesh allows users to share files and folder over multiple devices. The PaaS allows the users to set the runtime environment or modify the environment based on the requirement, for a specific application (Gonçalves & Ballon, 2011). The PaaS also provides the necessary programming environments, libraries, and tools for the users to allow them to create, manage, and deploy applications. For example, Google App Engine, Microsoft Azure, and Amazon Map are PaaS services currently available in the market. The IaaS provides

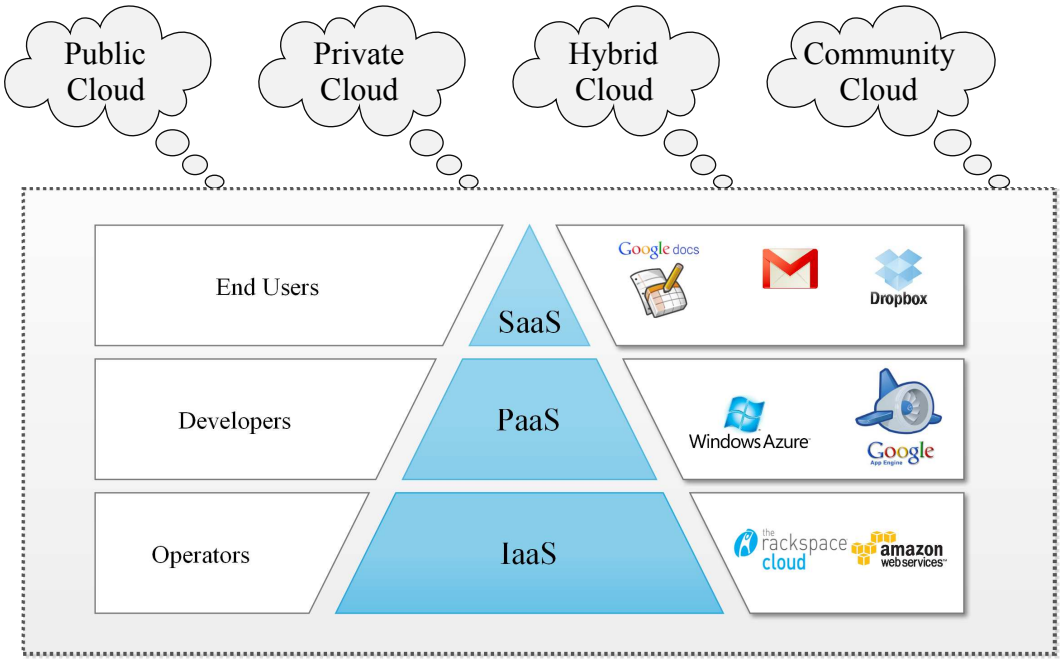


Figure 2.1: Different Service Delivery Models in Cloud Computing

a more flexible environment for the users by providing storage, computation, and networking infrastructure at the Virtual Machine (VM) level. For example, Amazon Elastic Cloud Computing (EC2) and Simple Storage Service (S3) are two of the IaaS services (Jing, Ali, She, & Zhong, 2013; Subashini & Kavitha, 2011; Takabi, Joshi, & Gail-Joon, 2010). Figure 2.1 illustrates three models in cloud services: SaaS,PaaS, and IaaS.

2.2.2 Security Background

The fundamental concept of security regarding to outsource data storage is composed of three components: confidentiality, integrity and availability (CIA Triad). Confidentiality refers to ensuring that unauthorized persons do not have privilege to access the information. The main idea behind the data confidentiality in cloud and mobile cloud computing is to encrypt the data before transferring them by using Attribute-Based Encryption (ABE) (Sahai & Waters, 2005), Proxy re-encryption (PRE) (Blaze & Strauss, 1998), or Role-based access control (RBAC) (Sandhu, Coyne, Feinstein, & Youman, 1996). Integrity ensures that unauthorized persons will not able to alter the stored data in the cloud by using remote data checking methods. Finally, data availability in cloud ensures that

the information is accessible anytime anywhere. This section introduces commonly used cryptosystems for data integrity and their modes of operation in cloud computing.

2.2.2.1 Fundamentals of Cryptography

Encryption algorithms are important procedures of cryptography that are used to preserve integrity and confidentiality of data. There are two types of encryption schemes: symmetric and asymmetric encryption schemes. In the symmetric encryption scheme, the sender and receiver need to establish a secure communication session based on a share key and also use the same key for encrypt and decrypt the message. Therefore, this type of encryption method will not be applied for two parties who never met before.

The main disadvantage of symmetric encryption scheme is that the encryptor to communicate with different persons requires storing the different key for each person. However, the generation and management of the large number of keys is complex and needs a large storage space. Common symmetric key encryption algorithms include Advanced Encryption Standard (AES) (Daemen & Rijmen, 2000), Data Encryption Standard (DES), Triple-Data Encryption Standard (3DES) (Chaum & Evertse, 1986),and One- and Snow (Ekdahl & Johansson, 2003). In the Asymmetric algorithms, each user has two keys as public and private keys which are used to establish a secure session of communication across a network without requiring a symmetric key. Although this scheme is more secure than their symmetric counterparts, the symmetric encryption scheme is faster than it.

2.2.2.2 Probabilistic encryption

The most cryptographic algorithms are deterministic which means that every plaintext will always be encrypted in the same ciphertext under a fixed encryption key. The adversaries may be able to use this feature to compute some partial information about the plaintext or the encryption key. For example, in RSA cryptosystem, the relation between

one bit of ciphertext and plaintext, that is called Jacobi symbol, is predictable (Fontaine & Galand, 2007).The probabilistic encryption is proposed to address this issue in symmetric and asymmetric cryptosystem. In the symmetric schemes, a unique random vector is computed for each plaintext by using the randomized methods to generate different ciphertexts with a same key. Since the security analysis in asymmetric schemes is mathematical and formal, their randomized methods have to be analyzable in the deterministic schemes, for example in (ElGamal, 1985; Goldwasser & Micali, 1982).

2.2.2.3 Homomorphic encryption

As mentioned earlier, with the growth in communication networks and the mobile devices and their increasing capabilities over the last few years, the demand for storing sensitive data on the cloud storage and delegating computations to untrusted cloud has increased exponentially. Data encryption is a crucial method to store and access data securely in the cloud. However, the main issue is how to perform computation on the encrypted data without decrypting it and to obtain the same result as performing on the original data.

Rivest, Adleman, and Dertouzos (1978) was the first to overcome this issue by homomorphic encryption. During the last few years, the extensive researches have been carried out on this area and the application of this method has increased dramatically in cryptographic protocols such as, secure data outsourcing, secret sharing scheme, and multiparty computation. An encryption function $(E())$ is homomorphic if for any $E(x), E(y)$,and $E(x \odot y)$ can be computable without decrypting x and y for operation $\odot$:

$$\forall x,\, y \in M, E(x \odot y) \leftarrow E(x) \odot E(y) \tag{2.1}$$

Where M denotes the set of plaintext. This definition indicates that performing operation on plaintext before encryption is equal to perform operation on the corresponding

ciphertexts after encryption. One example of a deterministic multiplicatively homomorphic cryptosystem is RSA scheme which was created by Rivest, Shamir, and Adleman (1978) on $M = (Z/NZ, .)$ where N is product of two large prime number (p,q). If the public and privet keys are define as $\{K_e = (e, n), K_d = d|n = pq, ed \equiv 1 \, mod \, \phi(n)\}$, the encryption and decryption of a message is calculated by:

$$E_{K_e}(m) = m^e mod \, n, \tag{2.2}$$

$$D_{K_e}(m) = E_{K_e}(m)^d mod \, n, \tag{2.3}$$

Therefore, the encryption of the product of two messages is computed based on multiplying the corresponding ciphertexts as follow:

$$\left.\begin{array}{l} E_{K_e}(m_1) = {m_1}^e mod \, n, \\ E_{K_e}(m_2) = {m_2}^e mod \, n \end{array}\right\} \Rightarrow E_{K_e}(m_1.m_2) = E_{K_e}(m_1) \, .E_{K_e}(m_2), \tag{2.4}$$

The Paillier cryptosystem is another example of homomorphic cryptosystem that is proposed by Paillier (1999) based on RSA scheme on $M = (ZN2, ., +)$. The public and private keys in Paillier method are defended as $K_e = (N, g)$ and $K_d = lcm((p-1),(q-1))$, where *lcm* denotes lowest common multiple. The owner selects a random number (r) and encrypts the plaintext by:

$$E_{K_e}(m) = g^m {r^N} mod \, N^2 \tag{2.5}$$

If $E_{K_e}(m_1) = g^{m_1} {r_1}^N mod \, N^2$ and $E_{K_e}(m_2) = g^2 {r_2}^N mod \, N^2$, the product of two ciphertexts is calculated by the following formula:

$$E_{K_e}(m_1) \, .E_{K_e}(m_2) = g^{(m_1+m_2)}(r_1 r_2)^N mod \, N^2 = E(m_1 + m_2) \tag{2.6}$$

2.2.2.4 Applications and Properties of Homomorphic Encryption Schemes

Homomorphic encryption schemes have been widely applied in the different areas of cryptography because it allows manipulating an encrypted data without losing the security and privacy of data. In the following several applications and properties of Homomorphic cryptosystems are briefly reviewed.

- ***Protection of mobile agents:*** The homomorphic cryptosystem is able to ensure the security and privacy of mobile devices by encrypting the whole program or the communicated data because the architecture of most computers are constructed based on binary strings and only need multiplication and addition (Sander & Tschudin, 1998). There are two ways to protect the mobile agent based on homomorphic encryption: (a) computing with encrypted data in which such algorithm is used to work on encrypted data, and (b) computation with encrypted functions in which the homomorphic scheme is in charge of evaluating and preserving the security of an encrypted functions of mobile devices.

- ***Secure data access and sharing scheme in cloud computing:*** One of the important applications of homomorphic cryptosystem is to ensure the confidentiality of outsource data and provide secure data access and sharing in cloud computing. However, the most homomorphic based methods incur a huge computation and storage overhead on cloud and client parties.

- ***Digital Watermarking schemes:*** Standard watermark detection methods usually are constructed based on the symmetric or asymmetric cryptography which are vulnerable to several security risks. For example, accessing the watermark information and symmetric key leads to remove watermark completely, the knowledge of the public detection key in asymmetric watermarking schemes also increases threat of oracle attacks. One of the efficient ways to overcome these security issues is

to apply Zero-knowledge watermark detection based on homomorphic encryption (Adelsbach, Katzenbeisser, & Sadeghi, 2002).

- ***Zero-knowledge proofs:*** Zero-knowledge proof is a method to convince a next party that the statement is true without learning anything as a result of this process. This method is a theoretical application of homomorphic cryptosystems. Remote data checking in cloud computing is a crucial application of zero-knowledge proof where the cloud wants to show the client that the outsourced data is remain intact. In the next section some Zero-knowledge proof in cloud computing are reviewed.

- ***Commitment schemes:*** Commitment is an essential part of some modern cryptographic protocols including zero knowledge proofs and secure computation in which a player is able to select a value from some set and assures that he cannot change his value or statement. The selected value should be kept secret until he decides to reveal it for the other parties. Homomorphic cryptosystem provides an efficient way to implement some commitment schemes. For example, one particular application of commitment is in zero-knowledge proofs for two main purposes: (a) the prover uses the commitment schemes to ”cut and choose” proofs based on the verifier challenge and only disclose what should be related later in the proof (Goldreich, Micali, & Wigderson, 1991), and (b) commitment schemes is capable of preventing verifiers from specifying their choices ahead of time in a commitment and compose a parallel method without revealing additional information (Goldreich & Krawczyk, 1996).

2.2.2.5 Algebraic signatures

Algebraic signature is a type of hash functions with algebraic properties that allows to compute the signatures of unseen messages in a limited way. The fundamental feature of algebraic signature schemes is to take a signature of the sum of some random blocks

gives the same result as taking the sum of the signatures of the corresponding blocks (Schwarz & Miller, 2006).

Let an element γ in the Galois field composed of a vector of various non-zero elements $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n)$. An algebraic signature of file F including n data block $(f[1], f[2], \ldots, [f[n])$ is computed by:

$$S_\gamma(F) = \sum_{i=1}^{n} f[i].\gamma^{i-1} \tag{2.7}$$

In the following, a number of algebraic signature properties are listed.

- ***Proposition 1:*** Litwin and Schwarz (2004) also shown that the algebraic signature of concatenation of two blocks $b[1]$ with length r and $b[2]$ is computed by:

$$S_\gamma(f[i]||f[j]) = S_\gamma(f[i]) \oplus r^\gamma S_\gamma(f[j]) \tag{2.8}$$

- ***Proposition 2:*** The algebraic signature of summation of two files, F and G, is equal to summation of signature of the files.

$$S_\gamma(F+G) = S_\gamma(F) + S_\gamma(G) \tag{2.9}$$

Proof:The summation of signature of the two files, F and G consist of n blocks, can be computed by:

$$
\begin{aligned}
S_\gamma(F) + S_\gamma(G) &= \sum_{i=1}^{n} f[i].\gamma^{i-1} + \sum_{i=1}^{n} g[i].\gamma^{i-1} \\
&= \sum_{i=1}^{n} \gamma^{i-1}(f[i] + g[i]) \\
&= S_\gamma(F+G)
\end{aligned}
$$

In the rest of this chapter, the remote data auditing methods are studied critically.

2.3 Remote Data Auditing Technique

Today, most of the individuals and organizations are motivated to outsource the data to the cloud to reduce the cost and time involved in procurement and maintenance of local storage infrastructure. In cloud computing, the Cloud Service Provider (CSP) is in charge of managing the cloud storage services. Therefore, the Data Owners (DOs) lose the physical control over the data. Instead, the management of data is delegated to an untrusted third party. On the other hand, the CSP or any inside attackers are able to arbitrarily change the amount of stored data without any user knowledge or permission (B. Chen, Curtmola, Ateniese, & Burns, 2010). Therefore, several issues need to be resolved before storing the sensitive data in the cloud. For instance, how can the user completely put her trust in the CSP for preserving the outsourced data? Is it possible for the CSP or any inside attackers to arbitrarily change the amount of stored data without user knowledge or permission? Do the users have to download the whole outsourced data to check the integrity of them? Is there any way to update the outsourced data without having to download the entire data?

The Remote Data Auditing (RDA) refers to a sampling of the collected data in the cloud and evaluating the data with various criteria, such as validity, accuracy, and integrity as a way to verify the reliability of the storage provider (Cong et al., 2010). In this section,

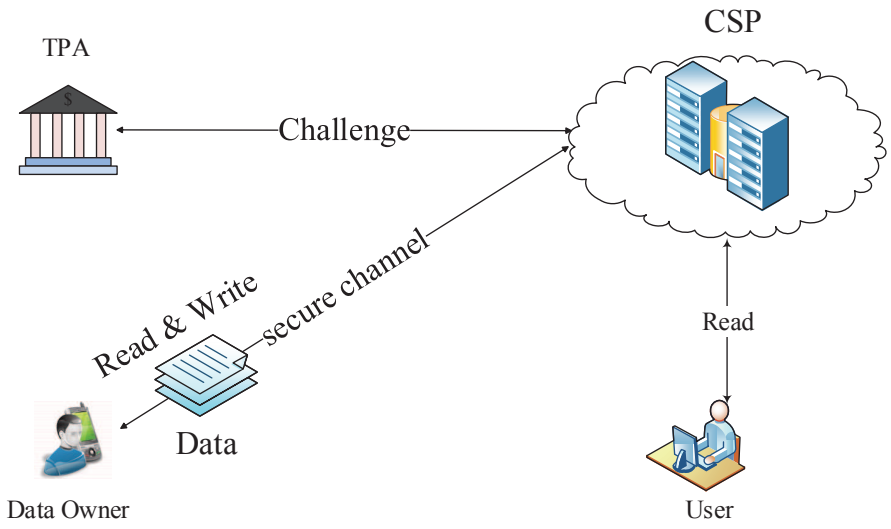


Figure 2.2: The network architecture of RDA in cloud computing

we detail the architecture of the RDA for the single cloud server and taxonomize the RDA requirements.

2.3.1 Remote Data Auditing Architecture

The RDA approaches for the single cloud servers usually include four main components, namely: (1) User: it represents an enterprise or individual having permission to read the stored data in the cloud, (2) DO: Enterprise or businesses which store their data in the cloud storage having the ability to do update operations (modify, delete, and insert), (3) CSP: This entity is responsible to back-up the user data and generates a proof as a response of the received challenges, and (4) Third Party Auditor (TPA): auditing the outsourced data and its verification is done by TPA. It actually ensures whether the data remains intact over the passage of time in public auditing models. Private auditing schemes, however, cannot support the TPA and DOs in order to check the integrity of the data (Sood, 2012; Sookhak, Talebian, Ahmed, Gani, & Khan, 2014; Sookhak et al., 2015). Figure 2.2 clearly depicts the typical RDA components and their interactions.

2.3.2 Taxonomy of Remote Data Auditing

Figure 2.3 shows the thematic taxonomy of remote data auditing in cloud computing that is categorized based on Security Requirements, Performance Metrics, Security Objectives, Auditing Modes, and Updating Modes.

The security requirements attribute indicates a number of properties which must be taken into account to propose a secure RDA method, as follows: (1) Robustness equips the auditing methods with mechanisms to mitigate arbitrary amount of data corruption (Ateniese et al., 2011), (2) Fairness ensures that a dishonest data owner is unable to access the data in the cloud storage and manipulate it (Zheng & Xu, 2012), (3) Data Deduplication ensures maximum use of available storage space by recognizing distinct chunks of data with identical content and eliminating redundant data. Considering that more than 75% of the outsourced data in the cloud are not unique, deduplication can dramatically reduce the required space to store a large data set (Gantz & Reinsel, 2010; Storer, Greenan, Long, & Miller, 2008), (4) Data Recovery allows users to recover small or large fraction of file corruptions outsourced to the cloud. This requirement can be achieved by using some methods such as forward error correcting code (FEC) (Clark & Cain, 1981), or Read-Salmon code (Lin & Costello, 2004), (5) Dependability protects the stored data against Byzantine failures (Castro & Liskov, 2002), malicious data modification, and server colluding attack to augment data availability, (6) Batch Auditing ensures that TPA is able to quickly manage multiple auditing tasks which are received simultaneously from different users and in a cost efficient way, and (7) Data Privacy ensures that the auditors should not be able to learn or guess the data content or have a copy of original data. In other words, data confidentiality should be preserved against the auditors (Wei et al., 2013).

The performance metrics attribute includes a set of important measures such as computation cost (processing time), communication cost, and storage cost, and probability of

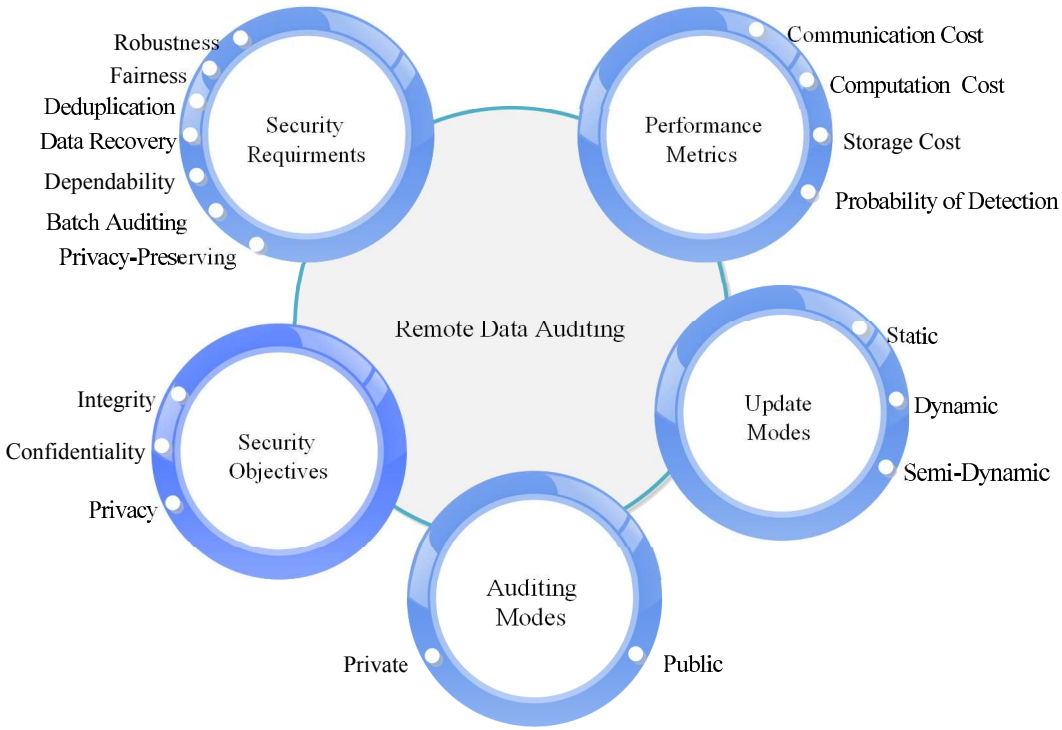


Figure 2.3: Taxonomy of Remote data auditing in cloud computing (Sookhak et al., 2014)

detection which are needed to be kept optimized when designing a RDA method. The implemented method requires incurring the least computation, communication and storage overhead over the client and server while the probability of detecting data corruption achieves the maximum value (Bowers, Juels, & Oprea, 2009; Oprea, Reiter, & Yang, 2005).

The security objective attributes indicate the RDA method is able to ensure which type of security components (integrity, confidentiality, and privacy). The attribute of the auditing mode shows that who is responsible for verifying the outsourced data. In a private verification, the DO only has to check the integrity of the data. However, in a public verification mode, the DO is able to delegate the auditing task to the trusted third party.

The next attribute is auditing mode including public and private auditing. In public auditing mode the integrity of outsourced data is checked by third party auditor (TPA) while in the private mode, the data owner is only able to audit the data.

The attribute of the uploading mode indicates the type of data modification that can be supported by the protocols. The current RDA methods employ three different strategies for updating the outsourced data blocks in the single cloud server. (1) In the static approach, the user must retrieve the data and upload the modified data on the cloud. This process imposes high computation and communication overheads on the cloud side and to the device side. (2) In the dynamic uploading approach, the user is able to update the stored data to the cloud by inserting, deleting, and modifying a portion of the file, or appending to the file remotely rather than downloading the entire file (Yang & Jia, 2012). (3) Semi-dynamic model: allows the owner to make partial update operations on the outsourced data (Sookhak, Talebian, et al., 2014).

the attribute of

2.4 The State-of-the-art Remote Data Auditing Approaches: Taxonomy

RDA is a crucial technique to check data integrity in the cloud and mobile cloud computing. According to identified security requirements in previous section, the state-of-the-art remote data auditing approaches are analysed and taxonomized into three models, namely Provable Data Possession-based (PDP-based), Proof Of Retrievability-based (POR-based), and Proof Of Ownership-based (POW-based) which are depicted in Figure 2.4. Also, few remote data auditing methods are described for each group and tabulated the comparison results in Table 2.1 and Table 2.2.

2.4.1 Provable Data Possession-based Methods

The first group of remote data auditing schemes in cloud and mobile cloud computing that is only responsible to preserve the integrity of outsourced data, is called Provable Data Possession (PDP). This type of methods usually include four main phases, such as: setup, challenge, proof, and verification. (1) Setup phase: in this phase, the input data is divided into n blocks and the unique tag (metadata) for each block is computed using

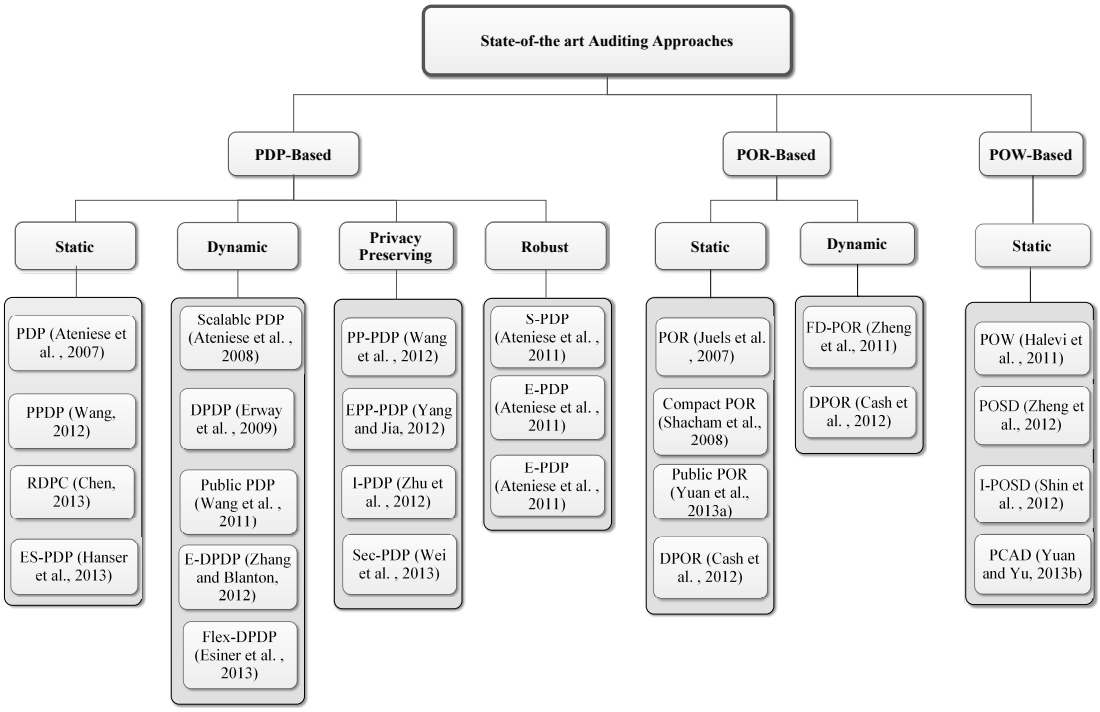


Figure 2.4: Taxonomy of State-of-the-art Remote Data Auditing Methods (Sookhak et al., 2014)

the distinctive formula. Finally, the input file and tags are sent to the cloud, (2) Challenge phase: in order to audit the cloud and check the correctness of the stored data, a verifier requires selecting some data blocks randomly as a challenge by using pseudo-random permutation, (3) Proof phase: upon receiving the challenge message, the prover generates a short integrity check over the received challenge message as a proof message - that usually includes the aggregation of the blocks and the tags – and sends it to verifier, (4) Verification phase: in the verification phase, the verifier validates the proof message regarding to the proof and challenge messages (Sookhak et al., 2015; Sookhak, Talebian, et al., 2014). The structure of PDP-based methods is shown in Figure 2.5.

In the rest of this section some PDP-based methods are critically reviewed along with their advantages and disadvantages.

2.4.1.1 Static PDP models

Ateniese et al. (2007) were the first to propose two provably-secure schemes by using the RSA-based Homomorphic Verifiable Tag (HVT) to verify the integrity of data storage

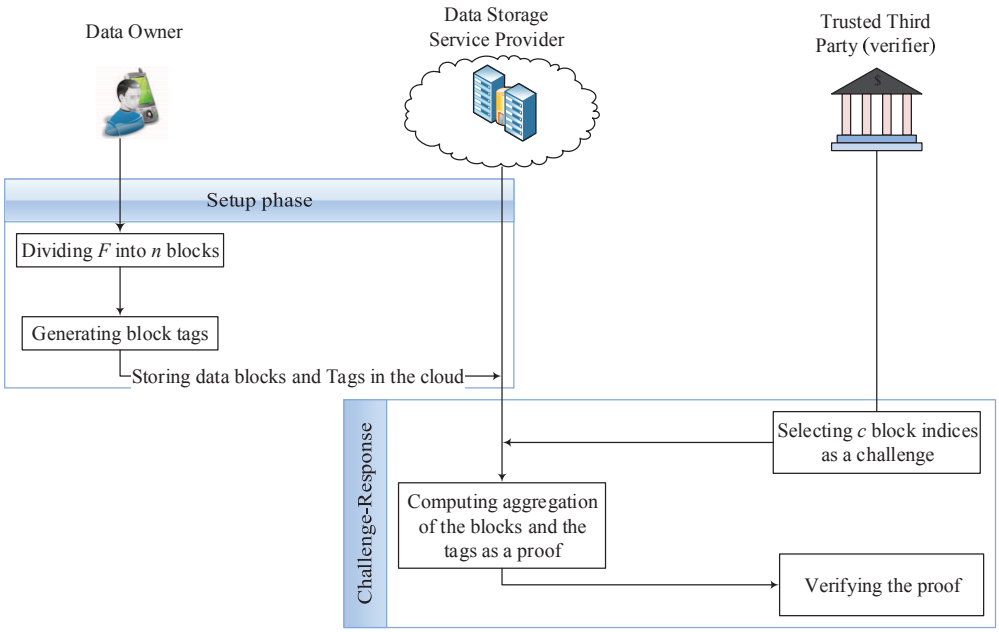


Figure 2.5: The structure of Provable Data Possession-based methods (Sookhak et al., 2014)

in the cloud without having to retrieve the data. HVT permits the client to check whether the server has certain blocks based on a proof that is constructed by the server, even when the client has no access to the blocks. Secure PDP (S-PDP) is the first model with strong guarantee on data possession by adding the robust feature to PDP based on spot-checking mechanism. Since that incur computation cost on the client, they suggested an Efficient PDP (E-PDP) to reduce the computation cost by assuring the possession of the combined blocks (Ateniese et al., 2011). However, these two schemes have several drawbacks such as: (1) incur expensive server computation and communication cost over the whole file because of using RSA numbering, (2) have linear storage for the user, and (3) fail to provide secure data possession completely when the prover has malicious intent.

Hanser and Slamanig (2013) offered a provable data possession method based on elliptic curves cryptosystem in which a data owner or third party are simultaneously able to audit outsourced data remotely. The main idea behind this method is generating a same tag for simultaneous private and public verifiability by identifying a vector for each data

block. Therefore, the input file including n data blocks, is represented by $\lceil \frac{n}{t} \rceil$ consecutive vectors where t is the length of each vector, as follows.

$$F = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_{\frac{n}{t}} \end{bmatrix} = \begin{bmatrix} F_{1,1} \dots F_{1,t} \\ \vdots \ddots \vdots \\ F_{\frac{n}{t},1} \dots F_{\frac{n}{t},t} \end{bmatrix} \tag{2.10}$$

Finally, the data owner calculates a tag (T_i) for each vector (F_i) by using hash function which maps to an elliptic curve group (Icart, 2009).

The data owner or TPA is always in charge of auditing the outsourced data in all PDP methods. In some situation, however, the client will be restricted due to unavailability of Internet connection, such as on the ocean-going vessel, in the jail or battlefield. On the other hand, the TPA is not able to perform remote data checking independently, when the data owner is not capable of passing the verification step. The TPA does not have permission to take the further countermeasures without informing the owner. H. Wang (2012) overcome this issue by proposing a Proxy PDP (PPDP) method by using the bilinear pairing technique in which a remote data integrity checking task is delegated to a proxy according to a warrant. As it is shown in Figure 2.6, the PPDP scheme consist of four steps: (1) the system parameters and the public keys are generated by TTP, (2) the warrant and corresponding sign are generated by the data owner to delegate the tasks to proxy, (3) the proxy verifies the received warrant and the corresponding Signature, (4) the client divides the input into n blocks, generates the corresponding tag for each block, and store them on the cloud, (5) the CSP validates the tags to resist the malicious clients, and (6) the proxy is able to audit the stored data in the cloud by using challenge-response method.

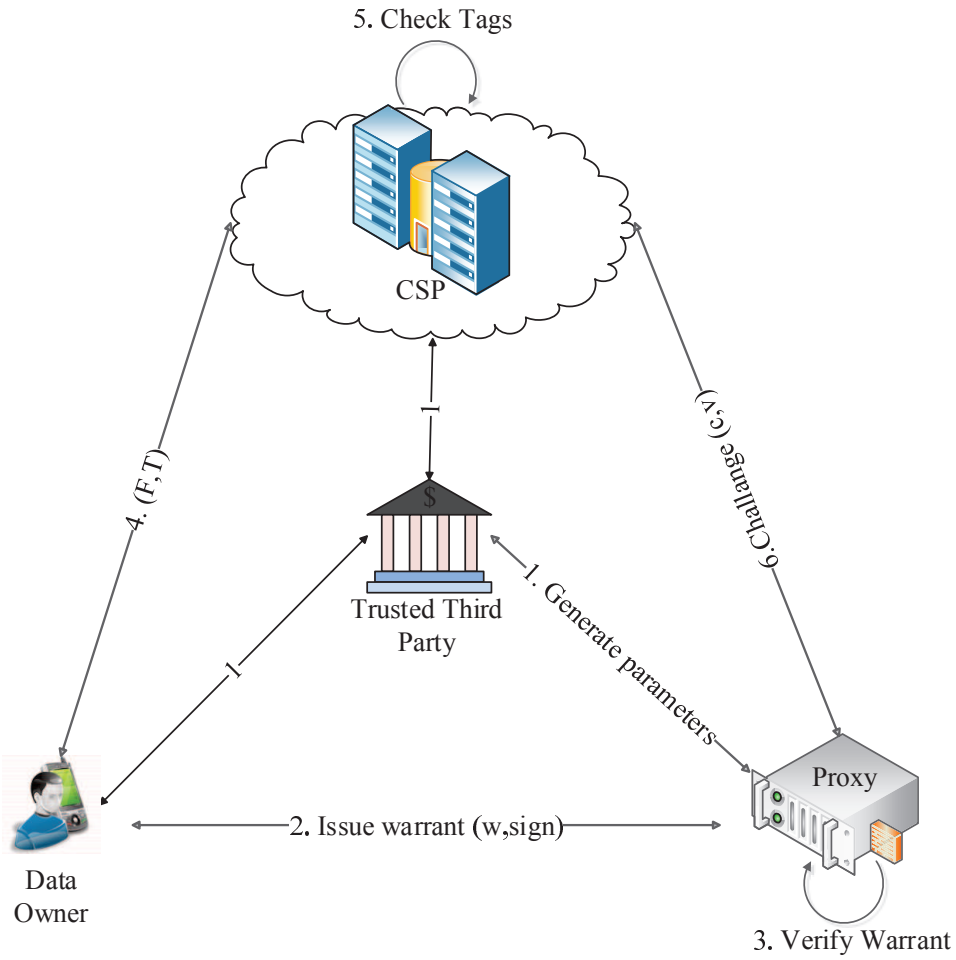


Figure 2.6: The architecture of Proxy Provable Data Possession method (Sookhak et al., 2014)

2.4.1.2 Dynamic PDP models

As mentioned in section 2.3, dynamic data update is a useful feature for data auditing methods that allows data owners to update their outsourced data whenever necessary without the need to download the data. The dynamic data update includes update, insert, append and delete operations.

Ateniese et al. (2008) considered the problem of static PDP methods for updating data and developed a new PDP protocol called Scalable PDP based on symmetric-key cryptography to solve the scalability, efficiency, and dynamic update issues in original PDP method. The distinction between Scalable PDP and original PDP is that a certain number of short possession verification tokens are precomputed by the data owner before

uploading data on the cloud. Each token are generated by using a random challenge and a random set of data blocks. Although the scalable PDP supports dynamic data, the update operations are limited to modify, delete, and append. To update a data block in the cloud, the data owner needs to retrieve all tokens and replace the hash of the block's old version with the new one by using XOR operation (Bellare, Guérin, & Rogaway, 1995). However, once an update operation is asked by user, it is required to re-compute all remaining tokens which are computationally expensive and thus impractical for large files. In addition, even though the scalable PDP enjoys more efficiency than original PDP, but the number of updates and challenges is restricted and it does not support public verifiability in which other parties rather than data owner also can check the integrity of outsourced data.

One of the effective ways to add dynamic data support to the current RDA protocols is making use of authenticated data structures such as Merkle Hash Tree (MHT) (Merkle, 1980), skip list (Pugh, 1990). The first fully dynamic PDP method is designed by (Erway et al., 2009) by combining of the original skip list (Pugh, 1990) with an authentication dictionary (Goodrich, Tamassia, & Schwerin, 2001) and rank-based information to enable efficient authentication of the clients' updates. In rank-based authentication skip list structure, each node stores the homomorphic block tag of the data block $(T(b[i]))$, level of node, the number of leaf nodes which are reachable from that node as a rank of node, searching variables, and a label of node. Erway et al.(2009) also proposed another dynamic PDP method by using Rank-based RSA trees to enhance the probability of detecting a tampered block in dynamic PDP method. The main difference of these two methods is storing the rank information trees on the internal nodes in Rank-based RSA scheme. To update a data block in Dynamic PDP, the client requests to retrieve the homomorphic block tag of this data block $(T(b[i]))$ and its proof. In delete operation, the client needs to access the homomorphic block tag of previous block $(T(b[i-1]))$ and its proof.

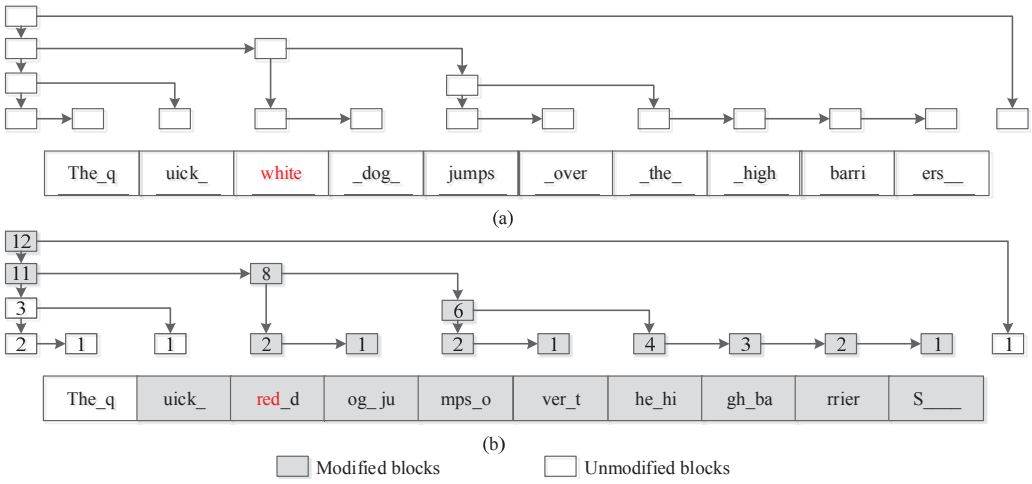


Figure 2.7: Rank-based Authenticated Skip List with block size 5, (a) before updating, (b) after updating (Sookhak et al., 2014)

In insert operation, the height of the tower of the skip list associated with the new block is also re-computed by the client. After verifying the proof, the client requires calculating the label of the start node of the skip list after the update by (Papamanthou & Tamassia, 2007). In the last step of update operations, the server updates the skip list on the basis of the received parameters.

Dynamic PDP (DPDP) method employs Rank-based Authenticated Skip List to efficiently support dynamic data update with $O(logn)$ complexity (Erway et al., 2009). However, the variable size of updated file incurs more overhead on other blocks with $O(n)$ complexity because the indices of the blocks are used in the skip list. For example, Figure 2.7 shows the outsourced file that is divided into some blocks. If the data owner decides to change the "white" to "red", it needs to balance the list by deleting the some blocks and insert them in the new place because the size of blocks in Rank-based Authenticated Skip List is fixed.

Esiner, Kachkeev, and Ozkasap (2013) overcome this issue by implementing a Flexible Length-based Authenticated Skip List method in which the indices of the bytes of the file are used to facilitate inserting, updating, deleting, or challenging a specific block consisting of the bytes at specific indices of the file. The significant advantage of FlexList

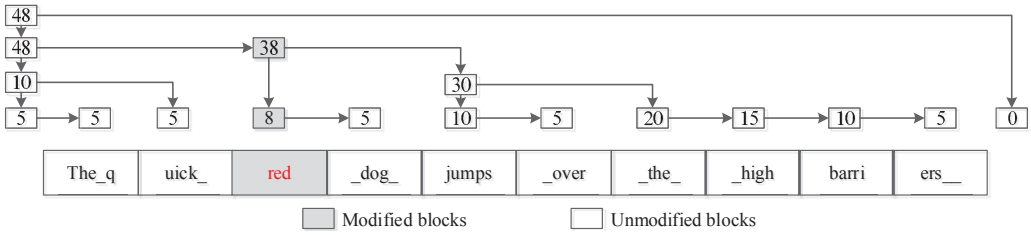


Figure 2.8: Updating operation in FlexList method (Sookhak et al., 2014)

method is its compatibility with the variable size of data block and data update. In other words, each node indicates how many bytes can be reached from this node instead of the number of accessible blocks. As a result, FlexList scheme is faster dynamic PDP in terms of data update with complexity where *u* is the size of update. Figure 2.8 shows that the data owner only needs to update the 3rd leaf of the list and the considering fathers.

Since prior dynamic data possession methods require verifying the cloud for each data block update operation, they incur high computation overhead on cloud and data owner. On the other hand, today, many web-based service providers include the web services, blogs, and other web-based application providers tend to outsource their data to the cloud or remote servers. In this way, users of such services are able to get access to the data anytime and from anywhere, and perform functions such as deleting, modifying, or inserting new data to the stored data, simultaneously. For instance, most of the popular blogs which are hosted by a cloud-based server permit their subscribers to delete, append, or remove blog content, freely. In this context, the data auditing methods should be able to manage multi-user access to the shared data on the cloud without leaking or losing data. Sometimes, the clients also need to retrieve previous versions of their data or they need to verify the integrity of the data without concerning about computation and storage overhead (Sookhak, Talebian, et al., 2014).

Y. Zhang and Blanton (2012) design an efficient dynamic provable possession based on a new data structure (that is called block update tree) to address these issues. The data owner and server needs to store the block update tree in order to overcome the requirement

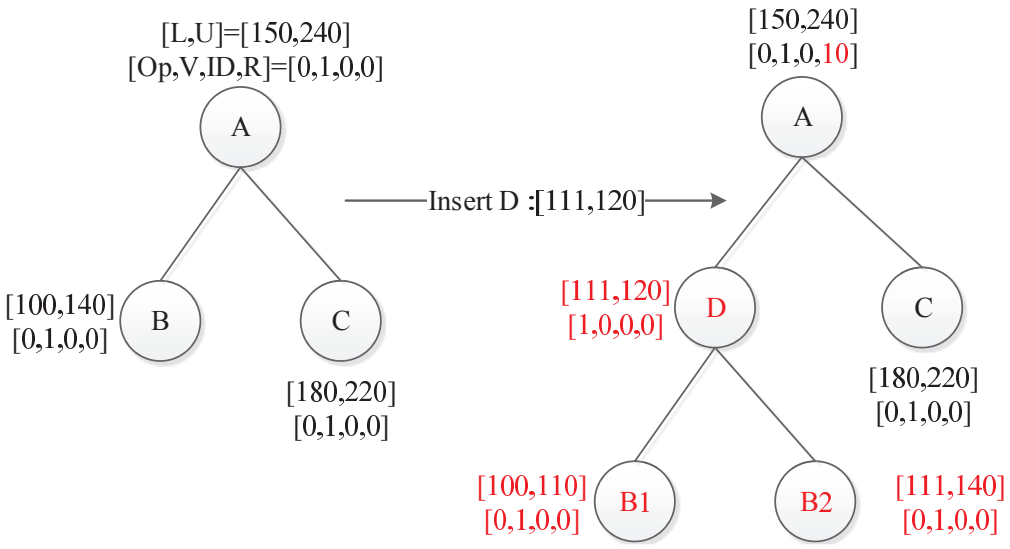


Figure 2.9: Insert operation in block update tree (Sookhak et al., 2014)

of verification for each dynamic operation. The main characteristic of this tree is that it is always balanced irrespective of the number and order of dynamic operations on the storage. Moreover, the size of the maintained update tree is independent of the outsourced data size because when the owner restores some data blocks in the cloud, all previous copies of the considering data blocks are deleted. The block update tree is a binary tree in which each node consists of some attributes such as: (1) node type (op) indicates the type of operation that is performed on the node $(delete = -1,\ modify = 0,\ insert = 1)$, (2) data block range (L,U) represent the range of data blocks in which the index of left child is always lower indices than L and the index of right child is always higher than U. As a result, some standard algorithms such as AVL tree (AdelsonVelskii & Landis, 1963) are able to be used to efficiently balance the tree. (3) offset (R) is used to identify the indices of data blocks after insert and delete operations, and (4) version number (V) represents the number of data block modification. For example, Figure 2.9 shows the node balancing function when a new node (D) is added to the tree. Since that the range of this node is $[111,120]$, it is inserted as a left child of A. The B also needs to be increased because of the range overlap between B and D.

MHT is a simple and effective model of the authentication structure which is pre-

sented as a binary hash tree. This data structure is used to detect any tampering and to prove that a set of element remains unaltered. The leaves of the MHT are the hashes of authentic data values and the other nodes are computed by hashing the combination of the hash of left and right children $(h(h(left\ child)\ ||h\ (right\ child)))$. However, MHT has a node balancing drawback which occurs after inserting or deleting a series of requests. As a result, this technique is not directly applicable in provable data possession schemes.

Q. A. Wang et al. (2011) proposed a Public Provable Data Possession (Public PDP) method by combining the MHT data structure (Merkle, 1980) and bilinear aggregate signature (Boneh, Gentry, Lynn, & Shacham, 2003) to address the node balancing issue in MHT. When data owner decides to update a data block, she sends the new data block along with its authentication tag to the cloud. Upon receiving the update request, the cloud performs the update operation, re-generates the root of the MHT, updates aggregation block tags and returns the signed root $(sign_{pr}(root\ of\ MHT))$. Finally, the owner validates the signed root to ensure the performance of update operation. Figure 2.10 illustrates the effect of insert and delete operations on the MHT in the Public PDP method.

2.4.1.3 Privacy-Preserving models

Cloud-based collaborative authoring is a fledgling service that helps the clients to share private documents with the others anywhere and anytime. The cloud-based structure of collaborative authoring service augments usability and fault tolerance; and achieves efficient resource allocation and global accessibility (Sheng-Cheng, Wu-Hsiao, Ming-Yang, & Chun-Yuen, 2012). However, by storing data to a remote server, the clients lose the physical control over data and delegate management of data to an untrusted cloud service provider. As a result, to protect the privacy of data, the clients need to encrypt the data using cryptographic techniques before outsourcing those data to the cloud (Kamara & Lauter, 2010). Since that data owner and the co-authors only have the encryption key

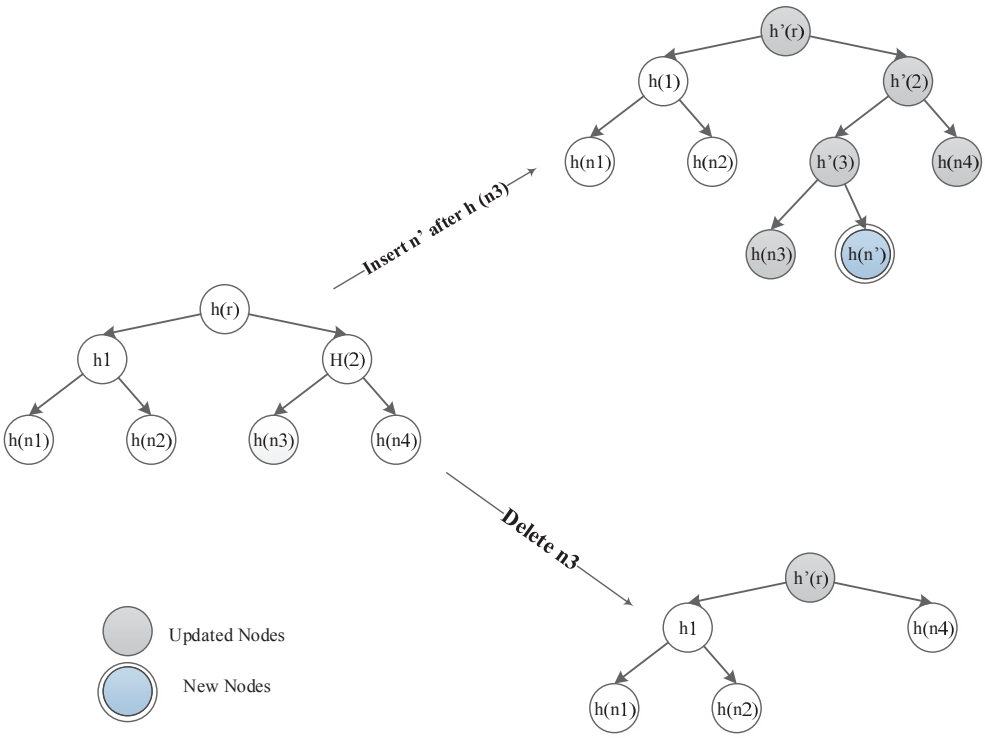


Figure 2.10: The effect of insert and delete operations on Merkle Hash Tree in Public PDP method (Sookhak et al., 2014)

in collaborative text editing services, unauthorized users are unable to access the shared document.

When the co-authors access to an outsourced document, the collaborative services are in charge of downloading the last version of the document and decrypt it using the appropriated key. On the other hand, by modifying a small part of the file, the owner and the co-authors need to encrypt and decrypt the whole file to obtain the last version of the file. Then, by increasing the size of documents or the number of authors, the required time to encrypt and decrypt the document is also increasing.

Yeh, Su, Chen, and Lin (2013) addressed this issue by proposing an efficient and secure cloud-based collaborative editing approach using the Red–Black tree to reduce the number of data that need to be encrypted. In other words, when an authorized user (the data owner or one of the co-authors) modifies a block of the file, instead of a whole file, the modified block only requires to be encrypted and updated. The Red-Black tree is

a type of binary search tree that was developed by Bayer (1972) as a symmetric binary B-trees to organize a part of text fragments or numbers. The main property of this data structure is that the computation order of search, insert, and delete operations is $O(logn)$, when n is the number of data blocks (nodes).

Data auditing methods usually assume TPA is a trustworthy agent and they neglect the privacy of data when TPA is involved. However, such assumption is illogical and leads to data leakage. C. Wang, Chow, Wang, Ren, and Lou (2012) considered this issue and proposed lightweight TPA protocol on the basis of public key under Homomorphic Linear Authenticator (HLA). The main idea behind of this method, namely privacy preserving PDP (PP-PDP) , is to integrate HLA with random masking technique to protect both data integrity and data privacy. In other words, before transferring the proof message to TPA, the aggregated blocks μ under challenge message needs to be blinded by using a random mask as follows:

$$\mu' = \mu + r.h((u^x)^r) \tag{2.11}$$

Where, μ' is the blinded blocks aggregation, μ is the aggregated blocks, r is a random mask, u is a cloud public key, x is a cloud private key, and h indicates the hash function.

Yang and Jia (2013) designed another privacy-preserving auditing for data storage security in cloud computing to address the storage overhead issue in Efficient privacy preserving PDP (EPP-PDP) (C. Wang et al., 2012). To this end, they use the Bilinearity property of the bilinear pairing and to generate an encrypted proof the challenge stamp such that the auditor is only able to verify the proof. They also improve the performance of this scheme by using the Data Fragment Technique and the HVT to reduce number of data tags, as follows: (1) the input data is divided into n blocks and each data block is

split to s sectors by using the data fragment technique, and (2) since that the number of sectors in all block must be same, the number of these sectors in the data blocks that have less sectors must be reached to s by appending the additional sectors with zero content. As a result, the number of data blocks in the input file (F) is calculated by:

$$n = \frac{sizeof(F)}{s.\log p} \tag{2.12}$$

where p is the size of each sector and one data tag is generated for s sectors. For example, when the size of each block is 20 Byte, 50 KB input file is divided to 2560 data blocks. Therefore, 2560 tags is needed to be generated for this file which incur 50 KB storage overhead while by using the data fragment technique, the storage overhead is reduced to 50/s KB. However, the main disadvantage of this method is that it is unable to efficiently support dynamic data update for large-scale files.

Zhu, Wang, Hu, Ahn, and Hu (2011) introduced a zero knowledge proof model to PDP in order to hinder data leakage during verification step. This method, that is called Improved PDP (I-PDP), also supports soundness property based on computation Diffie-Huffman assumption and the rewindable black-box knowledge extractor. The soundness property indicates that the cloud is not able to deceive the verifier to accept false statements. The principal idea behind this scheme is to randomize data blocks and their tags in order to prevent data and tag leakage during verification step.

Since that mobile computing devices have limited processing power, small storage capacity and short battery lifetime, the audit services are required to be efficiently designed for these devices. As a result, Yan, Hongxin, Gail-Joon, and Mengyang (2012) improved the performance of audit service in two ways: utilizes probabilistic query and periodic verification which helps to balance the computation and communication overhead. They also reduced the size of required storage using an algorithm which selects a

number of sectors for each block in the input file.

Wei et al. (2013) was the first to propose a privacy preserving and computation auditing by using Commitment-Based Sampling (CBS) technique (Du, Murugesan, & Jia, 2010) and designated verifier signature (Huang, Yang, Wong, & Susilo, 2011; J. Zhang & Mao, 2008) to achieve privacy cheating discouragement and minimize the computation cost in cloud computing. To store the input file on the cloud securely, the data owner splits the input file into *n* blocks and *n* storage space are allocated to the blocks by the CSP. Before transferring the data blocks to the cloud, each data block needs to be signed to enable the data auditing. After generating a secure communication tunnel by using a session key, the data blocks, and corresponding signature are transmitted to the cloud. When the data blocks are received, the CSP decrypts data blocks by using a session key and verifies the signature by using its secret key.

The main contribution of this method, namely Secure PDP (Sec-PDP), is to use the CBS technique based on Merkle Hash Tree (MHT) to provide computation security in two steps: (1) computation request step: in which the positions index of data blocks $(I = \{I_1, I_2, \ldots, I_n\})$ and a computation service request including a set of some basic functions $(F = \{f_1, f_2, \ldots, f_n\})$ such as data sum, average, and other complicated computations are submitted to the cloud. (2) commitment generation step: when the computation request is received by the cloud, the requested data blocks based on their position index set are retrieved and considering functions are computed on them $(y_i = f_i(b[i])$ and the value of intermediates nodes are computed by $V = H(V_{leftchild})||H(V_{Rightchild})$. Finally, the root of $MHT(R)$, its signature and the set of computation results are transferred to the user through transmission tunnel. As a result, the data owner or the TPA is able to verify the storage correctness and the computation correctness by using challenge-response method and re-building the MHT. Figure 2.11 shows the steps of this method to provide the security and privacy for storage and computation in cloud computing.

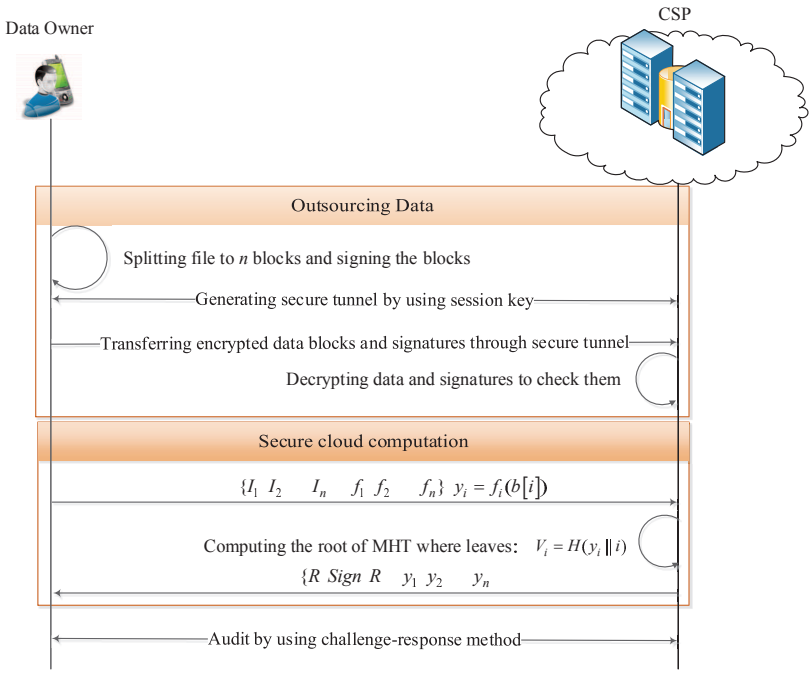


Figure 2.11: Security and privacy for storage and computation in cloud computing (Sookhak et al., 2014)

2.4.1.4 Robust Data auditing

The most of data auditing methods rely on selecting small portions of the data randomly as a challenge and checking their integrity is called spot checking. However, this technique is only able to detect a fraction of the data corruption in the server and the client cannot find corruption of small parts of the data. Ateniese et al. (2011) was the first to empower the PDP protocols to mitigate the arbitrary amount of data corruption (is called robust feature) by integrating Forward Error Checking (FEC) with PDP methods. In other words, the input file firstly needs to be encoded by using the FEC technique and then the encoded file is used as an input file to the PDP methods. There are different ways to encode the input file that causes the auditing schemes to include different properties and performance characteristics. The main distinction between these encoding techniques is related to the way of encryption or permutation the data blocks in each constraint group.

- ***Simple Reed-Solomon:*** to achieve this goal, the input file is divided into $k-$symbol chunks and then each chunk is expanded it to $n-$symbol codeword by applying a

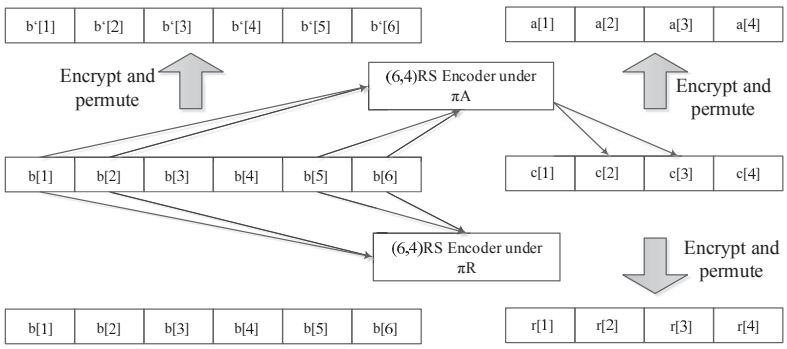


Figure 2.12: The $6,4)$ code permutation under (ΠR) and (ΠA) methods (Sookhak et al., 2014)

(n,k) Reed-Solomon code. The first k symbol in the output is the original file and the remaining symbols are parity blocks which are used to recover d erased blocks. The constraint group is defined concept as a group of blocks from the same encoding symbols (the original k blocks and their corresponding d parity). However, the attacker is able to manipulate the data by deleting a fixed number of blocks (any $d+1$ blocks of encrypted file from the same constraint group) due to the fixed number of k and d.

- ***Permutation All*** (ΠA)***:*** to overcome the first model, it requires hiding the constraints among blocks. As a result, all blocks of the encoded file should be permuted randomly. However, the resource-intensive nature of this method is slow because of performing permutation on all blocks. Furthermore, the robustness of the file is compromised by allowing sequential access to the original file.

- ***Permutation-Redundancy*** (ΠR)***:*** in spite of the Permutation all method (ΠA), only the parity symbols needs to be permuted and encrypted. The comparison of $(6,4)$ code permutation by the (ΠR) and (ΠA) methods are illustrated in Figure 2.12.

As mentioned earlier, the Reed-Solomon code is able to provide error correction in a static setting and it needs to hide the relationship between the symbols and the constraint groups by using permutation functions. On the contrary, since that dynamic data update

affects on the parity symbols of the corresponding group, the relationship between the symbols and the constraint groups has to be revealed in dynamic methods. Considering a contradiction between dynamic data update and robustness feature, an important question comes to mind: how is it possible to add the robust feature to dynamic provable data possession?

B. Chen and Curtmola (2012) solved this issue and introduced a Robust Dynamic PDP (RD-PDP) by presenting two new permutation functions such as Permute-Redundancy $(\Pi R - D)$, and Variable Length Constrain Group (VLGG).

- ***Permute-Redundancy*** $(\Pi R - D)$***:*** in this scheme, the (ΠR) technique is adopted to add the robustness feature to the dynamic PDP method. Since that the content of constraint group depends on the index of data symbol, to insert/delete a data block, the client has to download the whole file and re-compute the parity based on a new set of constrain blocks. However, to modify a data block, the client only needs to download the requested block and the considering parity. After updating the data block and computing the new parity symbol, the parity symbols have to permuted and re-encrypted to preserve the privacy of data against server.

- ***Variable Length Constrain Group (VLGG):*** to overcome the drawback of $\Pi R - D$ technique in insert/delete a data block, symbols are assigned to constraint groups on the basis of the content of symbols instead of the position of symbols in $\Pi R - D$. As a result, the data owner is able to update (insert and delete operations) the symbol, by only downloading the affected parity and updating the considering parity symbols of the considering constraint group.

They also combined Reed-Solomon codes (Reed & Solomon, 1960) with Cauchy Matrices (Plank & Xu, 2006) in order to reduce communication overhead of RS and

support efficient dynamic updates. In addition, the Cauchy Reed Solomon codes are two times faster than classical form of Reed Solomon.

2.4.2 Proofs of Retrievability-based methods

Proof Of Retrievability (POR) is a type of cryptographic Proof Of Knowledge (POK) to ensure the privacy and integrity of outsourced data in the untrusted cloud without having to download the files. It also provides data recovery and mitigates data corruption by performing Forward Error-correcting Codes (FECs) in which the verifier has capability to recover the file when a considerable fraction of the file is uncorrupted, as proved by spot-checks. The main difference between POR and PDP is the security features which they provide, because in the POR approach the client's data are completely stored on the server, while the PDP-based methods only guarantee that most of the client's data are kept in the server and a small portion of the data may be lost by the server. In addition, the POR method stores a redundant encoding of the client data on the server (Cash, Küpçü, & Wichs, 2012).

2.4.2.1 Static POR methods

The first POR method was proposed by Juels and Kaliski Jr. (2007) on the basis of sentinel blocks (called sentinel) which are concealed among other data blocks before transferring to the cloud. The sentinel blocks are computed by using a one-way function (f) as follows:

$$sentinles : \{s_1, s_2, ..., s_w\} \rightarrow s_i = f(key, i) \tag{2.13}$$

Since modifying part of data affects sentinel blocks with a certain probability, the verifier only requires checking whether a random set of sentinel blocks is intact. The main disadvantage of POR scheme is that the number of challenges is limited by the

number of embedded sentinel blocks in to file.

Shacham and Waters (2008) designed another POR scheme, namely Compact Proof Of Retrievability (Compact POR), to improve the efficiency and security of POR protocol and overcome its limitation in terms of number of challenge. The Compact POR method relies on applying the BLS homomorphic signature (Boneh, Lynn, & Shacham, 2004) to aggregate the tags and generate a single short tag as a proof to minimize the network computation overhead (for *t* challenges) during checking the integrity of blocks. The authors used the Reed-Solomon code (Plank & Xu, 2006) to support the error recovery in two ways, such as public verifiability, and private verifiability. The main difference between these two methods is that in private verifiability model the verifier needs to know the DO's private key in order to validate the proof message rather than the DO's public key in public verifiability model.

Non-trivial (linear or quadratic) communication complexity is another crucial issue of the existing POR schemes which are designed based on the homomorphic commitment schemes because of the linear sizes of their proves. Yuan and Yu (2013a) proposed a new POR method (that is called Public Proof Of Retrievability (Public POR))with constant communication cost by using a constant size polynomial commitment technique (Kate, Zaverucha, & Goldberg, 2010). A polynomial commitment scheme helps the prover to generate a polynomial with a short string as a proof which can be used to audit the cloud. Since the polynomial commitments have a constant size and the overhead of opening a proof is constant, this scheme decreases the communication cost in POR scheme. In order to reduce the complexity of the challenge message of Public POR method, the authors follow the challenge technique in (Dodis, Vadhan, & Wichs, 2009) in which a Hitter sampler technique (Goldreich, 1997) is used to randomly select the indices of input file as a challenge.

2.4.2.2 Dynamic POR models

The main limitation of most the POR methods is to fail to support dynamic data update efficiently as the server is not able to distinguish the relation between the data blocks and encrypted codewords. Cash et al. (2012) overcome this difficulty and implement a Dynamic Proof Of Retrievability (Dynamic POR) by combining the POR scheme and Oblivious Random Access Machine (ORAM) technique (Goldreich & Ostrovsky, 1996; Goodrich, Mitzenmacher, Ohrimenko, & Tamassia, 2012). ORAM is a hierarchical data structure which allows the client to read and write from/to the outsource data in a private way by hiding the location of the codewords. This data structure includes several levels of hash tables to hold encrypted address-value pairs while the lower tables have more capabilities. The top tables store the most recently accessed data and the bottom tables keep the least recently used data. When the client wants to read a data block, the address of data is hashed to checks the proper location in the top table. If the data is found, some random positions in the remaining tables are checked to hide the location of found data blocks. Otherwise, the next level table is checked to find the hashed address again. After finding the data by checking the tables, the found data needs to be written into the top table. The write operation is easier than read in which the encrypted address and value are stored in the top table.

Figure2.13 shows the construction of Dynamic POR method, the data owner divides the input file into l blocks with size *k* and then applies the systematic Reed-Solomon code to each block for ensuring the data recovery on each block. In the last step, the data owner uses the ORAM technique to outsource the data and keep a short local state.

Fairness is another inherent security issue in dynamic POR methods in which a dishonest data owner legally accuses the truthful cloud service provider for tampering its outsourced data. Zheng and Xu (2011) addressed this issue by proposing a Fair-Dynamic

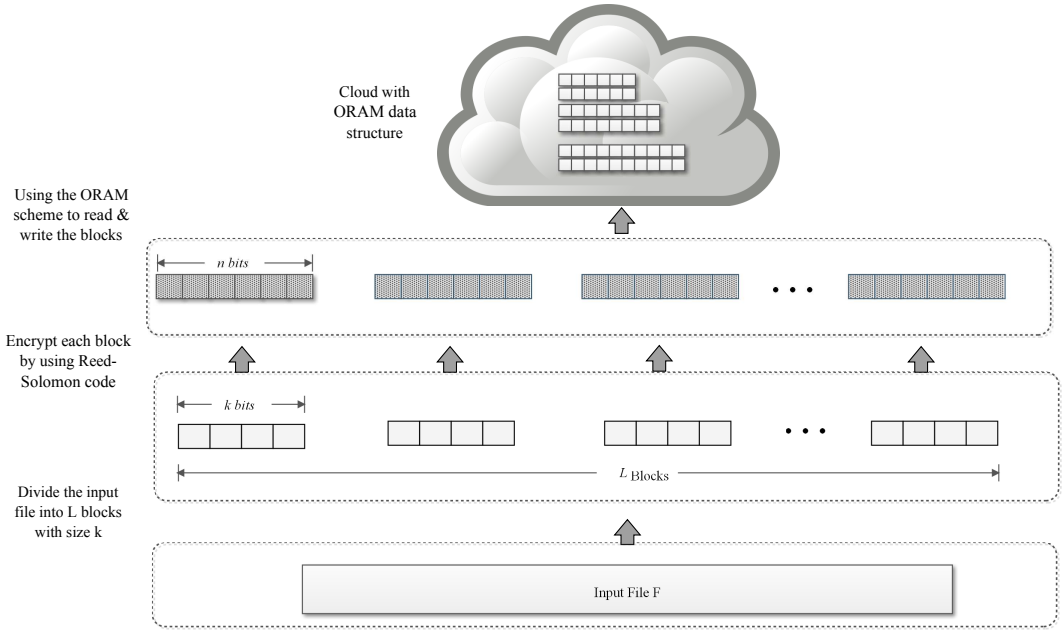


Figure 2.13: The structure of Dynamic Proofs of Retrievability via Oblivious RAM (Sookhak et al., 2014)

Proof Of Retrievability (FD-POR) method on the basis of 2-3 range based tree (rb23Tree) and Hash-Compress-and-Sign (HCS). Rb23Tree is an authenticated data structure based on a 2-3tree in which all intermediate nodes have two or three children and the leaves should be located on the same height. This tree is applied for authenticating a specific value that is stored at a specific leaf. The main feature of this tree is that leaf removal and leaf insertion involve logarithmic complexity. However, balancing rb23Tree is an expensive task and imposes a huge computational burden on the server.

The core idea underlying the FD-POR approach is securing the block index when computing the authentication tag in the compact POR scheme to prevent data leakage. As it is shown in Figure 2.14, the FD-POR scheme consists of four steps. In the first step, an input file is divided into n blocks ($F = \{f_1, f_2, ..., f_n\}$) and then each block is encrypted by using ECC ($F = \{f'_1, f'_2, ..., f'_n, f'_{n+1}\}$). In the next step, the encrypted blocks are hashed ($H_i = h(f'_1)$) to construct the rb23Tree with leaves $H_1, H_2, ..., H_n, H_{n+1}$. Finally, a flat tree is constructed to achieve the fairness.

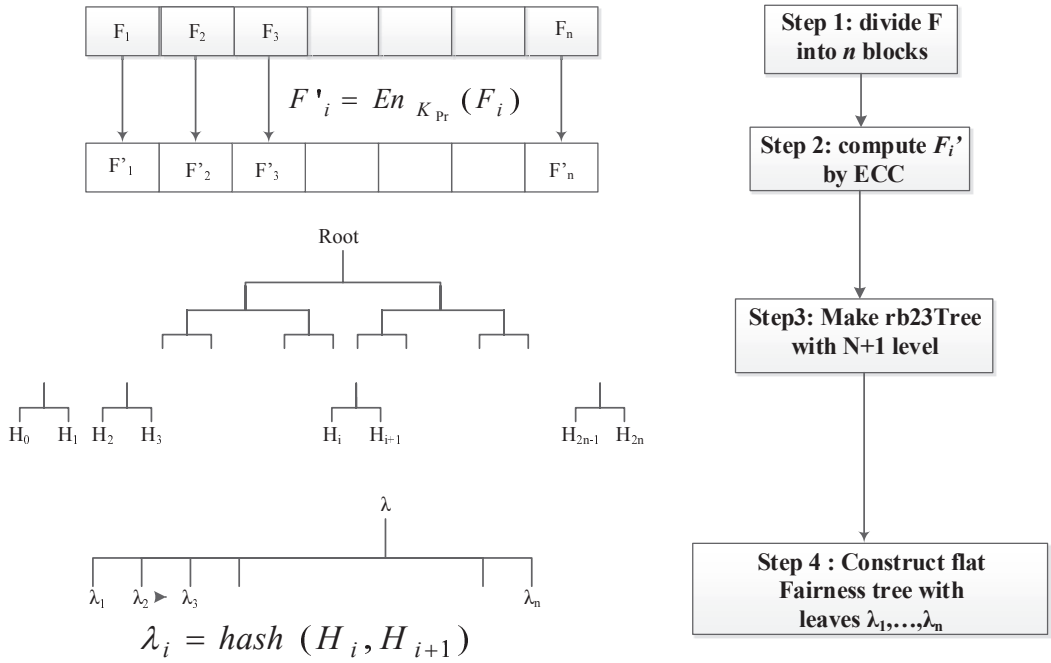


Figure 2.14: The main idea behind FD-POR scheme (Sookhak et al., 2014)

2.4.3 Proof of Ownership-based Methods

Data deduplication is a type of single-instance data storage (data compression) techniques, which is used to remove data redundancy and duplicate copies of data to provide a cost-efficient storage (Mandagere, Zhou, Smith, & Uttamchandani, 2008; Meyer & Bolosky, 2012). Deduplication techniques are responsible to recognize a common set of bytes within or between files (known as chunks) and allow the storing of a single instance of each chunk, irrespective of the number of repetitions. In typical storage system that support data deduplication, a client needs to convince the server of having a copy of the outsourced file by sending a hash of file to the server. If the server finds this hash in the database, accepts the client's claim and marks the client as the owner of that file; otherwise ask him to upload the entire file. However, this method is vulnerable against some security attacks, because anyone who gets the hash value is permitted to access the file (Halevi, Harnik, Pinkas, & Shulman-Peleg, 2011; Harnik, Pinkas, & Shulman-Peleg, 2010).

Halevi et al.(2011) considered these security issues and proposed a deduplication

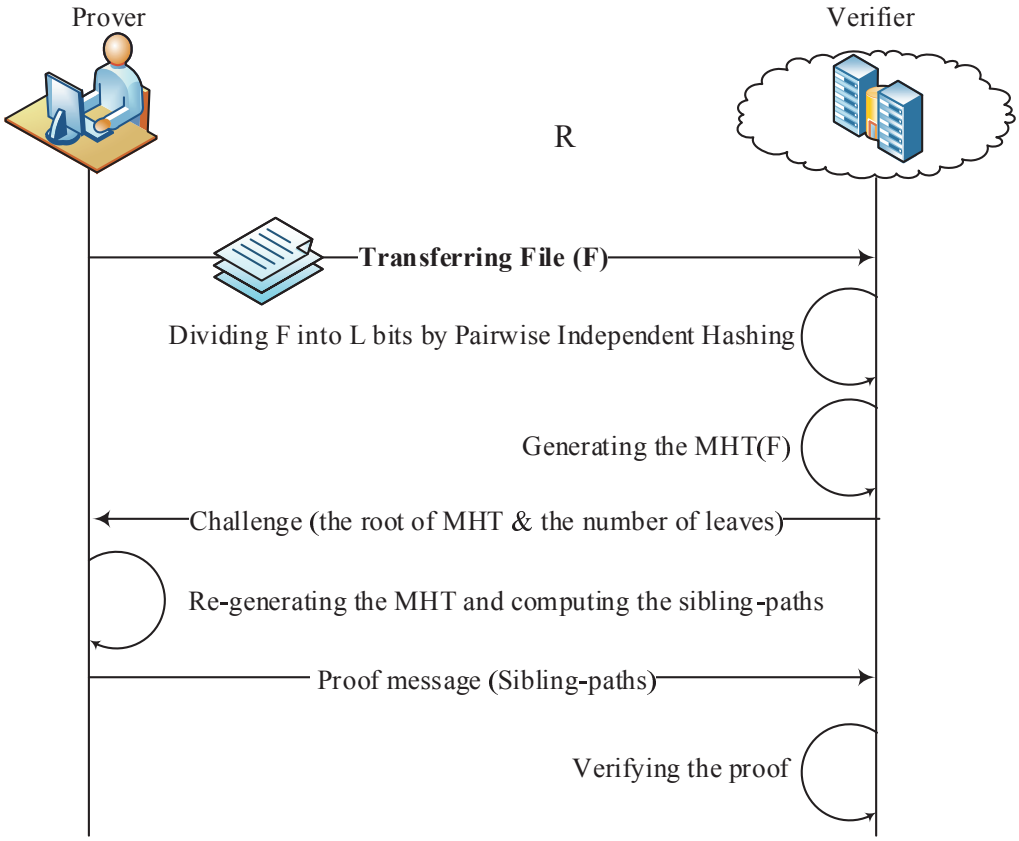


Figure 2.15: Proof of Ownership Protocol (Sookhak et al., 2014)

scheme in which the data owner is able to convince the server without transferring the file. This method (that is called Proof of Ownership (POW)) is constructed on the basis of Merkle Hash Tree (MHT) and Collision-Resistant Hash (CRH) functions. Since in the POW scheme the client needs to persuade the cloud, the role of prover and verifier is reversed. When the server receives the input file (F), he maps it to L bits using Pairwise Independent Hashing $(X = hPIH(F))$, and then constructs a Merkle tree based on it $(R = MHT\ (X))$. The verifier returns the root of the MHT and the number of its leaves $(RMHT\ ,\ nl)$ as the verification information. During the Proof step, the client computes the sibling-paths of all the leaves as a proof of deduplication and sends it to the cloud. Finally, the verifier validates the sibling-paths with regard to $MHT(X)$. Figure2.15 shows the core idea behind the Proof of ownership method.

Design a secure data deduplication method to provide simultaneously data auditing and deduplication in cloud computing is contradictory because security and efficiency

aspects seem to be two conflicting goals. In other words, deduplication eliminates the identical contents, while data security attempts to encrypt all the contents, and the same contents can then be encrypted by two different keys to create different ciphertexts (Marques & Costa, 2011; Storer et al., 2008).

Zheng and Xu(2012) introduced the first Proof of Storage with Deduplication (POSD) method by integrating two different concepts of Proof of Data Possession(PDP) and Proof of Ownership (POW) with the purpose of providing both security and efficiency. This method consists of four steps, such as key generation, uploading, auditing and deduplication. Before uploading the file to the cloud, the honest data owner is responsible to generate two pairs of public and private keys for checking data integrity and deduplication. The data owner divides the file into n block with m bits and then computes the block tags to upload to the cloud. The POSD method includes two challenge-response algorithms such as data auditing, which the data owner validate the correctness of data by sending challenge to the cloud as a prover, and deduplication, which the cloud verifies the claim of a client for having a copy of outsourced data.

The POSD scheme needs to meet the following security requirements: (1) server unforgeablity in which the server should provide a valid response to the client challenges with non-negligible probability, and (2) (k, Θ) uncheatablity in which dishonest clients are not able to deceive the server with non-negligible probability. The validity of the POSD scheme depends on the reliability of the clients in terms of performing key generation, while this assumption is not reasonable in the cross-multiple users and the cross-domain environment of cloud computing.

Shin, Hur, and Kim(2012)showed that dishonest client is able to manipulate the key generation step to create weak keys. As a result, this method fails to fulfill the unforgeablity and uncheatablity features as security requirements and it is vulnerable to some attack scenarios such as (1) Malware Distribution: a malicious client uploads a file in

the cloud and modifies it by attaching a malware to this file. This malware is distributed among clients when they execute the deduplication step to take the ownership of the original file, and (2) Unintended content distribution network: an adversary has capability to transfer the data to the other malicious client through the cloud by using the weak key. Shin et al.(2012) improved the security of POSD scheme by minimizing the client capability to control the key generation step. The core idea behind this method, namely Improved POSD (I-POSD) is to blind the keys with random values, when the server receive the data and corresponding tags.

The POSD scheme (Zheng & Xu, 2012) also suffers from the linear communication and computational cost on the client side regarding to the number of elements in each data block and the number of checking blocks during data auditing step. As a result, by increasing the number of mobile users, the communication and computational cost incur a huge overhead on the client who utilizes the resource constrained devices (e.g. Smart mobile phones) to access the cloud storage.

Yuan and Yu(2013b) proposed a new data storage auditing with deduplication capability to address the linear communication and computational cost on POSD scheme based on polynomial-based authentication tags and homomorphic linear authenticators. The communication complexity of auditing step in this method, that is called Public Cloud Auditing with Deduplication (PCAD), depends on transferring a challenge message, the aggregation tags of data blocks and the proof information which impose $O(1)$ as a total communication complexity on the client. However, the total communication complexity in POSD method is $O(s+k)$ because the CSP needs to transfer k authentication tags of the challenging blocks and s aggregated data blocks to the verifier, where s is the size of data block.

2.5 Comparison of Remote Data Auditing Schemes

This section compares the current remote data auditing protocols on the basis of the taxonomy presented in Figure 2.3. The commonalities and differences in such protocols are compared based on the presented parameters in such taxonomy. The comparison parameters considered are: Scheme Nature (SN), Protocol Type (PT), security pattern, Cryptography Model (CM), Batch Auditing (BA), Public Auditing (PA), dependability (Dep), and Data Recovery (DR). Table 2.5 shows a comparison of remote data auditing protocols based on such parameters and the assumptions and drawbacks of each method, as well.

The attribute of the SN indicates the various types of RDA techniques single cloud server that are divided in to provable data possession-based, proofs of retrievability-based, proof of ownership-based methods.

The attribute of the security pattern indicates the cryptographic algorithms that are used at the client side or the cloud side dynamically to audit or store data. The following security patterns are practiced to audit the outsourced data in the current data storage security schemes:

- ***Homomorphic encryption:*** as aforementioned in section 2.2.2.3, the homomorphic encryption is one of the main cryptographic techniques that is used to perform computation on the ciphertext without requiring to decrypt it. In single cloud server, the homomorphic encryption mechanisms are categorized into the following six types:

    1. *RSA Homomorphic*: this technique allows the client to combine the computed tags for multiple blocks of each file into a single value.
    2. *Paillier Homomorphic*: It is a type of RSA homomorphic cryptographic system in which the encryption of summation of blocks is equal to multiplication

Table 2.1: Comparison of Remote Data Auditing Protocols on the basis of The Basic Parameters

SN	PT	Protocols	Security Pattern		CM	BA	PA	Dep	DR	Assumptions and Drawbacks
			Data Auditing	Dynamic structure						
PDP-Based	Static	PDP (Ateniese et al., 2007)	RSA base HVT	Not support	Random Oracle	No	Yes	No	No	High computation and communication overhead on the server Fail to provide secure data possession completely Leak of information
		ES-PDP (Hanser & Slamanig, 2013)	Elliptic curves cryptography, HVT	Not support	Random Oracle	No	Yes	No	No	Only provides a probabilistic guarantee of possession
		PPDP (H. Wang, 2012)	Bilinear pairing technique	Not support	Random Oracle	No	No	No	No	Leak of information
	Dynamic	Scalable PDP (Ateniese et al., 2008)	Symmetric Key Cryptography	XMACC	Random Oracle	No	No	No	No	The number of queries is restricted Cannot provide a dynamic update fully. Only provides a probabilistic guarantee of possession
		DPDP (I) (Erway et al., 2009)	Homomorphic block tag	Rank-based Authentication Skip Lists	Standard	No	No	Yes	No	Cannot support data privacy Only provides a probabilistic guarantee of possession
		DPDP (II) (Erway et al., 2009)	Homomorphic block tag	Rank-based RSA Tree	Standard	No	No	Yes	No	Higher probability of detection but incurs more computation overhead than D-PDP (I)
		Flex-DPDP (Esiner et al., 2013)	Homomorphic block tag	Flexible Length-Based Authentication Skip List	Standard	No	No	Yes	No	Cannot support data privacy
		E-DPDP (Y. Zhang & Blanton, 2012)	Homomorphic block tag	Block update tree	Standard	No	No	Yes	No	Cannot support data privacy
		Public PDP (Q. A. Wang et al., 2011)	HLA	Merkle Hash Tree	Random Oracle	Yes	Yes	Yes	No	Leaks the data content to the auditor Incurs heavy computation cost of the auditor
	Privacy-Preserving	PP-PDP (C. Wang et al., 2012)	HLA with Random Masking	Merkle Hash Tree	Random Oracle	Yes	Yes	Yes	No	Incurs a heavy storage overhead on the server because of the large number of data tags
		EPP-PDP (K. Yang & X. Jia, 2012)	Bilinear pairing technique, HVT	Index Table	Random Oracle	Yes	Yes	No	No	The Index Table incurs huge storage overhead on auditor when the number of file increasing
		I-PDP (Y. Zhu et al., 2012)	Polynomial-time Rewindable, CDH	Not support	Standard	No	Yes	No	No	Support static update TPA is reliable
		Sec-PDP (Wei et al., 2013)	Commitment-Based Sampling (CBS), Verifier signature	Not support	Random Oracle	Yes	Yes	No	No	Support static update
	Robust	S-PDP (Ateniese et al., 2011)	RSA base HVT	Not support	Random Oracle	No	Yes	No	Yes	High computation overhead
		E-PDP (Ateniese et al., 2011)	RSA base HVT	Not support	Random Oracle	No	Yes	No	Yes	High computation overhead
		RD-PDP (B. Chen & Curtmola, 2012)	RS codes-based Cauchy Matrices	Rank-based Authentication Skip Lists	Standard	No	No	Yes	Yes	High computation overhead

SN: Scheme Nature , PT: Protocol Type, CM: Cryptography Model, BA: Batch Auditing, Public Auditing, Dep: Dependability, DR:Data Recovery

SN	PT	Protocols	Security Pattern		CM	BA	PA	Dep	DR	Assumptions and Drawbacks
			Data Auditing	Dynamic structure						
POR-Based	Static	POR (Juels & Burton S. Kaliski, 2007)	Sentinel-Based, RS codes	Not support	Standard	No	No	No	Yes	Limited number of challenges The computation and storage overheads, which arise from the error recovery and data encryption processes
		Compact POR(I) (Shacham & Waters, 2008)	HLA, Pseudo-random Function	Not support	Standard	No	No	No	Yes	Not applicable on IPS, Only applicable to private auditing
		Compact POR(II) (Shacham & Waters, 2008)	HLA based BLS Signature, CDH	Not support	Random Oracle	No	Yes	No	Yes	Not applicable on IPS
		Public POR (Yuan & Yu, 2013a)	Polynomial commitment scheme, CDH	Not support	Standard	No	Yes	No	Yes	The efficiency of this method is only approved in static environment.
	Dynamic	DPOR (Cash et al., 2012)	Sentinel-Based	ORAM Technique	Standard	No	No	No	Yes	The data owner cannot delegate the audit task to third party.
		FD-POR (Q. Zheng & Xu, 2011)	Hash-compress-and-sign	Rb23Tree	Random Oracle	No	Yes	No	Yes	Re-balancing and restructuring operations on range-based 2-3 tree during dynamic update phase
POW-Based	Static	POW (Halevi et al., 2011)	MHT, CRH	Not support	Standard	No	No	No	No	Cannot Support Integrity
		POSD (Qingji Zheng & Xu, 2012)	CDH	Not support	Random Oracle	No	Yes	No	No	Validity of key generation Linear communication and computational cost on the client
		I-POSD (Shin et al., 2012)	CDH, Random Key Blender	Not support	Random Oracle	No	Yes	No	No	Incurs huge communication and computational cost on the client
		PCAD (Yuan & Yu, 2013b)	Polynomial-based authentication tags,, HLA	Not support	Cryptography Model	Yes	Yes	No	No	Support static update Cannot support privacy

SN: Scheme Nature , PT: Protocol Type, CM: Cryptography Model, BA: Batch Auditing, Public Auditing, Dep: Dependability, DR:Data Recovery

of encryption of individual blocks (Paillier, 1999).

3. *Paillier Homomorphic*:The Homomorphic Verification Token (HVT) is a mechanism which can be used in distributed clouds to check the integrity and identify the errors by using the universal hash function (Carter & Wegman, 1979) and Reed-Solomon method (Plank & Ding, 2005) (e.g., (Ateniese et al., 2011, 2007; Hanser & Slamanig, 2013)).

4. *Homomorphic Linear Authentication (HLA)* utilizes a linear combination of the individual data block to generate a single value. Because the HLA uses a relatively small-sized BLS signature, it incurs less computation overhead than HVT (e.g., (Shacham & Waters, 2008; Q. A. Wang et al., 2011; C. Wang,

Chow, et al., 2012)).

5. *Homomorphic Verification Response (HVR)* is a mechanism that can be used to verify the integrity in distributed cloud storage by combining multiple responses from several clouds into a single value.

6. *Homomorphic MAC* cryptosystem is used to protect the scheme against pollution attack when the client and server have a shared secret key.

- (n,k)–***Reed Solomon (RS) code***: is one of the most important encryption mechanisms to correct block-based error codes, and is applicable in a wide range of digital communications and storage domain. The RS code encrypts k blocks to n block by adding $d = n-k$ bits parity to the original file to enable the DO to correct up to $\lfloor \frac{k}{2} \rfloor$ symbols.

- ***Symmetric key cryptography***: In the symmetric encryption scheme, the sender and receiver must establish a secure communication session based on a shared key and also use the same key to encrypt and decrypt the message (Dutta, Barua, & Sarkar, 2004).

- ***Pairing-based cryptography***: The main idea behind this method is to construct a mapping between the elements of two cryptographic groups for building a cryptosystem on the basis of the reduction of one problem in one group. It can be included Cryptographic Bilinear Pairings and Diffie-Hellman assumption (Dutta et al., 2004).

- ***Polynomial commitment scheme***: Helps a committer to commit to a polynomial with a short string that can be utilized by a client to verify claimed evaluations of the committed polynomial.

- ***Computational Diffie-Hellman (CDH)***: is a valuable assumption for cryptographic purposes and relates to the difficulty of computing the discrete logarithm problem within a cyclic group (Bao, Deng, & Zhu, 2003). This means that CDH is concerned with the mathematical operations that are completed quickly, but difficult to reverse.

The second part of security pattern includes some data structures that are used to provide dynamic data update, which are explained as follows:

- ***Rank-based skip list***: is an authentication model that enables the client to efficiently perform the update, delete, and insert operation on the outsourced data. Each of the node in this data structure stores the homomorphic block tag of the data block $(T(b[i]))$, level of the node, the number of leaf nodes that are reachable from that node as a rank of a node, searching variables, and a label of the node.

- ***Range-based 2-3 Tree (rb23Tree)***: is an authenticated data structure based on a 2-3 tree in which all nodes have two or three children, except the leaves which are located on the same height. This tree is applied for authenticating a specific value that is stored at a specific leaf (Zheng & Xu, 2011).

- ***Merkle Hash Tree (MHT)***: is a binary tree structure often used for data integrity verification. In the MHT, the leaves are the hash values of the individual blocks while the remaining nodes are calculated on the basis of the hash value for a combination of the two children nodes (Q. A. Wang et al., 2011). Figure 2.16 shows a MHT data structure for a file including four data blocks.

- ***Update Trees***: This hash tree has three main features which make it different from the other authenticated data structures such as (i) it is always balanced, (ii) its size is independent of the number and order of dynamic operations or outsourced data

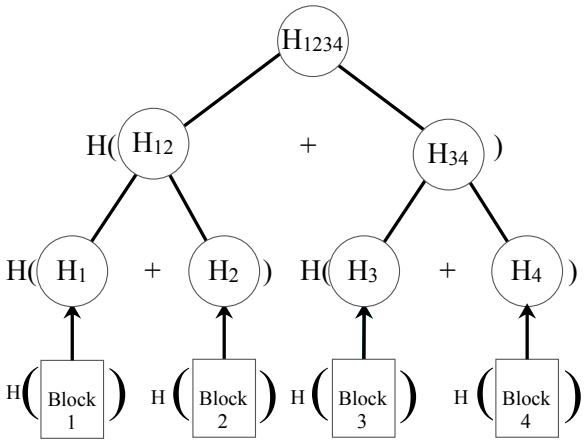


Figure 2.16: Merkle Hash Tree Structure

blocks on the storage, and (iii) each node in such tree is consist of a range of block indices instead of a certain index in MHT (Y. Zhang & Blanton, 2012).

The attribute of the cryptography model indicates a common methodology to design the cryptographic protocols, including: ***(1) Standard model:*** In this model of computation, the complex assumptions, such as hash functions (MD5, SHA) are used to prove the security of the scheme. However, achieving the security proofs in such a model is very difficult. ***(2) Random oracle model:*** The hash functions in such a model are replaced by a set of truly random functions to prove the security of the ideal system. It is noteworthy to mention that whenever a method is secure using the random oracle model, the implementation of such a system is also secure in the standard model (Canetti, Goldreich, & Halevi, 2004).

To analyze the efficiency of the remote data auditing approaches, there are several metrics that should be considered: ***(1) Computational cost (Processing time):*** Data auditing approaches impose different computation overhead on the client as a verifier and cloud service provider as a prover on the basis of their cryptographic algorithms. Client computation indicates the computational resources that are used by the client to generate the challenge and verify the proof message while server computation denotes the computation resources that the server uses to process an update step or compute a proof for

a block. ***(2) Communication cost*** shows the size of the challenge message sent to the prover and the size of proof message received by the verifier. There are two techniques to reduce computation and communication complexity in remote data auditing methods, such as sampling and batch auditing. In the sampling technique, the input file is divided into several blocks and a random number of blocks is used to perform batch processing (Erway et al., 2009). The batch auditing technique decreases the size of the proof message by sending a linear combination of random blocks to decrease the communication overhead (Q. Wang et al., 2009). ***(3) The probability of detection:*** is the last factor which represents the probability of detecting a cloud server's misbehavior. The comparison of the efficiency between some protocols based on the computation cost, communication cost, and the probability of misbehavior detection are represented in Table 2.5, where n is the number of blocks of each file, s is the number of sectors of a block, m indicates the number of symbols of a block, t shows the number of blocks that will be changed, c is the number of cloud service providers in multi-cloud, ρ and ρ_k are the probability of block corruption in a cloud server and k^{th} server in the multi-cloud, and $\Omega(.)$ is the proof size in the hash function.

2.6 Open issues and challenges

In this Section, we highlight some of the most important issues and challenges in deploying and utilizing the remote data storage auditing approaches as the future research directions.

2.6.1 Lightweight data auditing approach for mobile cloud computing

Developing lightweight remote data auditing approaches to improve the security of mobile users without any further limitation and requirement is a significant challenge in mobile cloud computing environment. Dividing the huge files into some blocks, generating the specific tag for each block, computing a challenge, and verifying the proof

Table 2.2: Efficiency comparison between some remote data auditing protocols

Protocols	Client computation	Server computation	Communication complexity	Probability of detection
PDP (Ateniese et al., 2007)	$O(t)$	$O(t)$	$O(mn)$	$1-(1-\rho)^t$
Scalable PDP (Ateniese et al., 2008)	$O(t)$	$O(t)$	$O(t)$	$1-(1-\rho)^{mt}$
DPDP(I) (Erway et al., 2009)	$O(t\log n)$	$O(t\log n)$	$O(t\log n)$	$1-(1-\rho)^t$
DPDP(II) (Erway et al., 2009)	$O(t\log n)$	$O(n^{\varepsilon}\log n)$	$O(t\log n)$	$1-(1-\rho)^{\Omega(\log n)}$
R-DPDP(VLCG) (B. Chen & Curtmola, 2012)	$O(t\log n)$	$O(t\log n)$	$O(\log^2 n)$	N/A
Public PDP (Q. A. Wang et al., 2011)	$O(t\log n)$	$O(t\log n)$	$O(t\log n)$	$1-(1-\rho)^t$
PP-PDP (C. Wang et al., 2012)	$O(t\log n)$	$O(t\log n)$	$O(t\log n)$	$1-(1-\rho)^t$
EPP-PDP (K. Yang & X. Jia, 2012)	$O(t)$	$O(ts)$	$O(t)$	$1-(1-\rho)^{ts}$
I-PDP (Y. Zhu et al., 2012)	$O(t+s)$	$O(ts)$	$O(t+s)$	$1-(1-\rho)^{ts}$
RDPC (L. Chen, 2013)	$O(t)$	$O(t)$	$O(t)$	N/A
POR (Juels & Burton S. Kaliski, 2007)	$O(t)$	$O(t)$	$O(mn)$	N/A
Compact POR(I) (Shacham & Waters, 2008)	$O(t)$	$O(t)$	$O(mn)$	$1-(1-\rho)^t$
Compact POR(II) (Shacham & Waters, 2008)	$O(t+s)$	$O(t+s)$	$O(s)$	$1-(1-\rho)^{mt}$
DPOR (Cash et al., 2012)	$O(t\log^2 n)$	$O(t^2\log^2 n)$	$O(t^2\log^2 n)$	N/A
POSD (Qingji Zheng & Xu, 2012)	$O(t)$	$O(n)$	$O((m+t)n)$	N/A

message are particular tasks in data auditing mechanisms that noticeably increase overall execution time and decrease the lifetime of resource constrained devices such as smart phones and tablets.

A feasible approach to decrease the side effect of remote data auditing approach on

mobile devices is to utilize the efficient public verification approach. As a result, the mobile user delegates the challenge and verification steps to the trusted third party to release the mobile devices from the communication and computation overhead of these steps. However, some of remote data auditing approaches (e.g. DPOR (Cash et al., 2012)) are unable to support the public verification technique. On the other hand, when the security of data is very important, data owners prefer to use the private verification methods. Therefore, implementing lightweight remote data auditing methods demands lightweight computation and communication techniques to be applicable on mobile devices.

2.6.2 Dynamic data update

To update a single bit of data or changes a bit location of the outsourced data in static data auditing methods, the data owner has to download the whole data, change the location of more than half bits of files, and upload it again to the cloud which incurs high communication overhead on the data owner. Therefore, dynamic data update is a vital feature to almost all remote data auditing approaches in the cloud and mobile cloud computing due to the dynamic nature of the data involved, such as electronic documents and log files. However, one of the main limitations of POR-based and POW-based methods is the need to support dynamic data update (Erway et al., 2009). Although Cash et al. (2012) proposed a first POR-based method to overcome the dynamic data update issue in cloud computing, the lack of public verification feature makes this method impractical for mobile cloud computing.

On the other hand, current dynamic data update methods also impose high storage and computation overhead on data owner (especially on resource restriction devices) resulting from re-balancing the stored tree, increasing the size of the tree. The main approach to address this issue is to develop an update tree with independent size of the outsourced data and without having to balancing procedure. As a result, enabling mobile

users to efficiently and dynamically update their outsourced data requires future research and developments (Sookhak, Talebian, et al., 2014; Sookhak et al., 2015).

2.6.3 Data access control over shared data

Today, many web-based service providers including the web services, blogs, and other web-based application providers tend to outsource their data to the cloud or remote servers. It is clear that users of such services require gaining access to their data anytime, anywhere, and simultaneously performing updating operations such as deleting, modifying or inserting new data to the stored data. For instance, most of the popular blogs which are hosted by a cloud-based server permit their subscribers to delete, append, or remove blog content, freely. However, most of the methods which are designed to ensure data integrity are unable to fulfill these requirements completely or will incur extra computation and high storage overhead on the users.

Data deduplication is a necessary feature of remote data checking mechanisms which has a noticeable effect on data communication and communication cost over cloud user. Though the POW-based approaches permit multi users to get access to the shared data, the users are unable to write, delete or modify the data in the same time. In this context, the remote data auditing methods need to manage multi-user access to the shared data on the cloud without leaking or losing data.

2.6.4 Data computational integrity

Recently, many data owners tend to outsource an arbitrary computation service to a cloud service provider (Cachin, 2011). Ensuring the integrity outsourced computations enable a client to transfer a computation step of remote data checking in PDP-based, POR-based and POW-based method to another computer and then, without executing the computation, merely checks the integrity of computation in the new computer (Setty, Blumberg, & Walfish, 2011). In other words, the practical and unconditional verifica-

tion is capable of contributing significantly to the development of remote data auditing approaches by reducing the computational overhead. However, current data integrity methods are unable to support data computation integrity as well. Migrating computational functions along with data into the cloud and using challenge-response approach to verify computation and data integrity can be a possible way to address this issue. This technique, which is useful for resource constraint devices to reduce the computation cost still needs certain degree of attention.

2.7 Conclusion

This chapter explains the concept of cloud computing, mobile cloud computing and discussed the different techniques used to ensure data integrity and privacy in the cloud and mobile cloud computing. It analyzes the current RDA methods in detail with the aim of highlighting the similarities and differences in the thematic taxonomy based on various parameters. It discusses the issues in the state-of-the-art RDA approaches and focused on the challenges pertaining to the security requirements to provide optimal and lightweight security frameworks. Several open challenges particularly, lightweight data auditing, dynamic data update, data access control, and computational integrity were presented as the most important open challenges for future efforts.

Auditing outsourced data in cloud computing is an emerging research area, which has been getting more attention in recent years. Current RDA approaches accomplish data checking process in diverse modes. Several approaches only audit the integrity of outsourced data, while a number of these approaches focus on error recovery and the rest of approaches are able to check the data ownership as well. Two different types of verification pattern are used, in private verification only the data owner is able to check the integrity of data instead of the TPA in the public verification mode. The ultimate goal of RDA is to preserve the integrity and privacy of outsourced data and computation in

single and distributed cloud servers regardless of underlying resource restrictions.

Current RDA methods employ a homomorphic cryptosystem to verify the integrity of the outsourced data, which incurs additional processing time and data transmission process. As a result, additional computation and communication cost arises in the verification phase of such methods. Moreover, current RDA methods use different types of hash trees to support dynamic data update. However, such data structure suffer from node balancing issue result in additional computation cost on the auditor.

Hence, current RDA methods requires considerable processing time for performing data integrity dynamically. In the next chapter, the problem statement of this research is analyzed by using the emulation environment. For example, It can be seen that when the size of outsourced file is large, the existing data structures are unable to efficiently support dynamic data update operations.

CHAPTER 3

PROBLEM ANALYSIS

3.1 Introduction

This chapter investigates the required processing time and transmission processes to verify the proof and perform data update operation for normal and large scale files in the traditional remote data auditing methods. The main objective of this chapter is to analyze the aforementioned research problem in section 1.3. The measurement parameters to analyze and establish the problem consists of processing time of data integrity, processing time of dynamic data update for normal and large scale files, data communication cost of dynamic data update, and processing time for frequent data update.

The problem analysis is carried out on the basis of the traditional integrity-based remote data auditing methods, due to the scope of the research. As mentioned in section 2.4, the integrity-based remote data auditing methods are divided into two categories based on data update operations, such as static and dynamic. The first auditing method that is selected for analysis is the Provable Data Possession (PDP) method (Ateniese et al., 2011, 2007), which is known as a fundamental of static method in this area and has relatively better performance than other methods as described in Table 2.2. Moreover, most of integrity-based methods have been designed based on the PDP method. Section 2.4.1 explains that most of dynamic integrity-based method use the binary tree as a data structure to support dynamic data update operations. In other words, such data structures prevent the CSP to use the old version of data block as a proof message. Among such methods, the Public PDP method (Q. A. Wang et al., 2011; C. Wang, Wang, Ren, Cao, & Lou, 2012) is used in the problem analysis section. This is because the public PDP

shows better performance than other integrity-based dynamic methods as indicated in Table 2.2. Thereby, public PDP has become a widely used auditing method in domain and has attracted large number of citation in literature.

The processing time of traditional RDA methods (Ateniese et al., 2011, 2007; Q. A. Wang et al., 2011; C. Wang, Wang, et al., 2012) is analyzed for different size of the outsource files in remote cloud server nodes. On the basis of the literature, the size of normal file is in the range of 10 MB -50 MB (Q. A. Wang et al., 2011; C. Wang, Wang, et al., 2012). Moreover, most of the existing cloud storage applications allow the users to store documents from 10 MB to 50 MB, such as Google Drive (*Google Docs, Sheets, and Slides size limits*, 2015).

The remaining parts of this chapter is organized as follows. Section 3.2 investigates the additional processing time of data auditing in the existing schemes. Section 3.3 analyzes the additional transmission cost of dynamic data update in existing auditing schemes. Section 3.4 studies the processing time of dynamic data update in static auditing methods. Section 3.5 analyzes the effect of replay attack on static remote data auditing methods. Section 3.6 presents the additional processing time of dynamic data update of large-scale files in the traditional auditing schemes. The effect of frequent updates on processing time of recent auditing methods is described in Section 3.7. Finally, the conclusion of this chapter is presented in Section 3.7 with conclusive remarks.

3.2 Analysis of Processing Time of Traditional RDA Methods on the Auditor

Performing a data auditing technique incurs specific excessive amount of computation burden (processing time) to the both auditor and service provider based on the type of cryptographic algorithm used. According to the RDA structure, processing time of auditor includes the following costs:

(i) Processing time of setup step (PT_s): The computational cost (processing time) to

process a file and generating the tags;

(ii) Processing time of challenge step (PT_c): The computational cost (processing time) to select a certain number of blocks randomly as a challenge message;

(iii) Processing time of verification step (PT_v): the required processing time to verify the proof message.

Hence, the total processing time $(\sum PT)$ of the RDA methods -that is incurred on the auditor- is computed by Equation 3.1.

$$\sum PT = PT_s + PT_c + PT_v \tag{3.1}$$

Since the data owner needs to divide the input file into *n* blocks and generate tags only one time per file in the set up phase, most of the researchers exclude the processing time of setup phase from the total processing time of RDA methods (Q. Wang et al., 2009). In other words, in contrast to the other phases of RDA method, the setup phase will not be repeated in the consequent auditing steps and no further cost will be incurred on the auditors due to performing setup step. As a result, the total processing time is calculated from Equation 3.2.

$$\sum PT = PT_c + PT_v \tag{3.2}$$

The processing time of data integrity in the remote data auditing methods are illustrated in Figure 3.1.

In the following, the processing time of data integrity in traditional data auditing is analyzed on the basis of two steps- challenge, and verification.

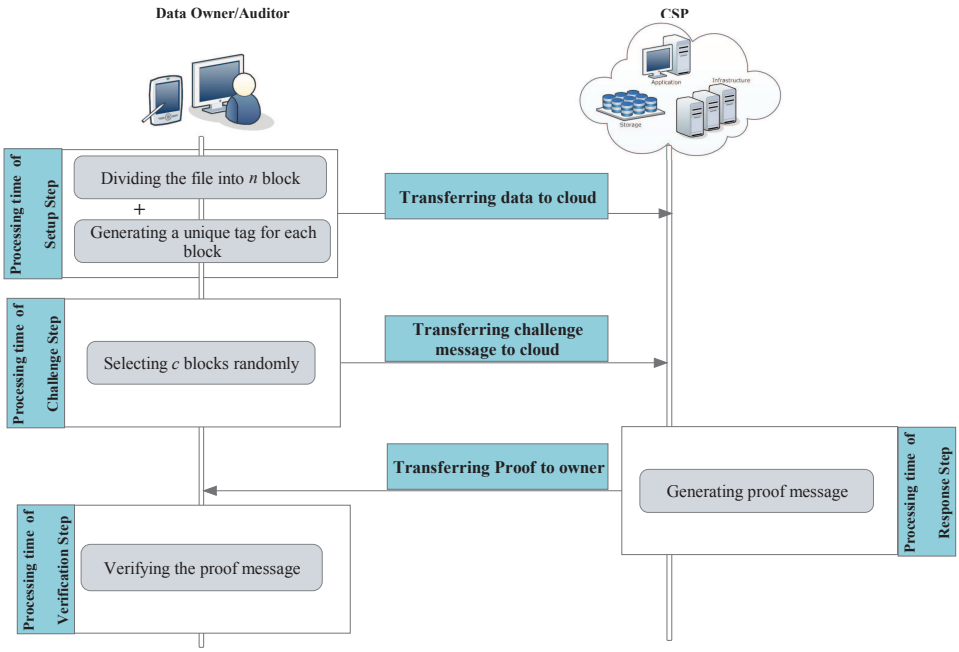


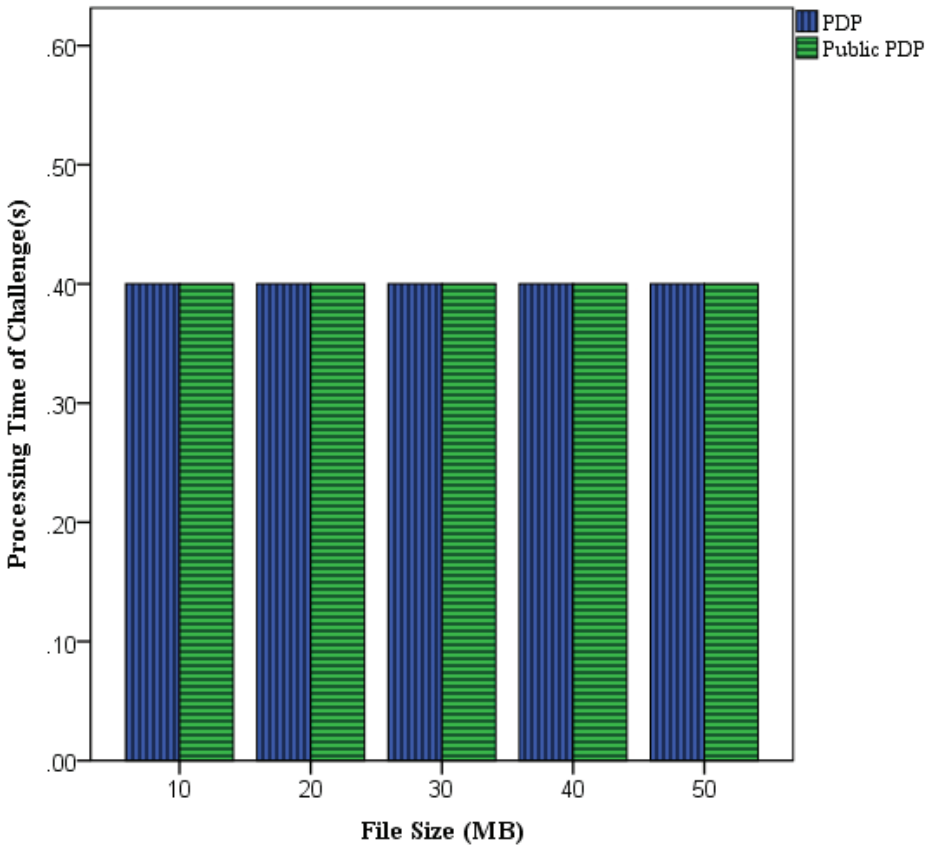
Figure 3.1: The Processing Time of Data Integrity in Remote Data Auditing Methods

3.2.1 Processing Time of Challenge Step

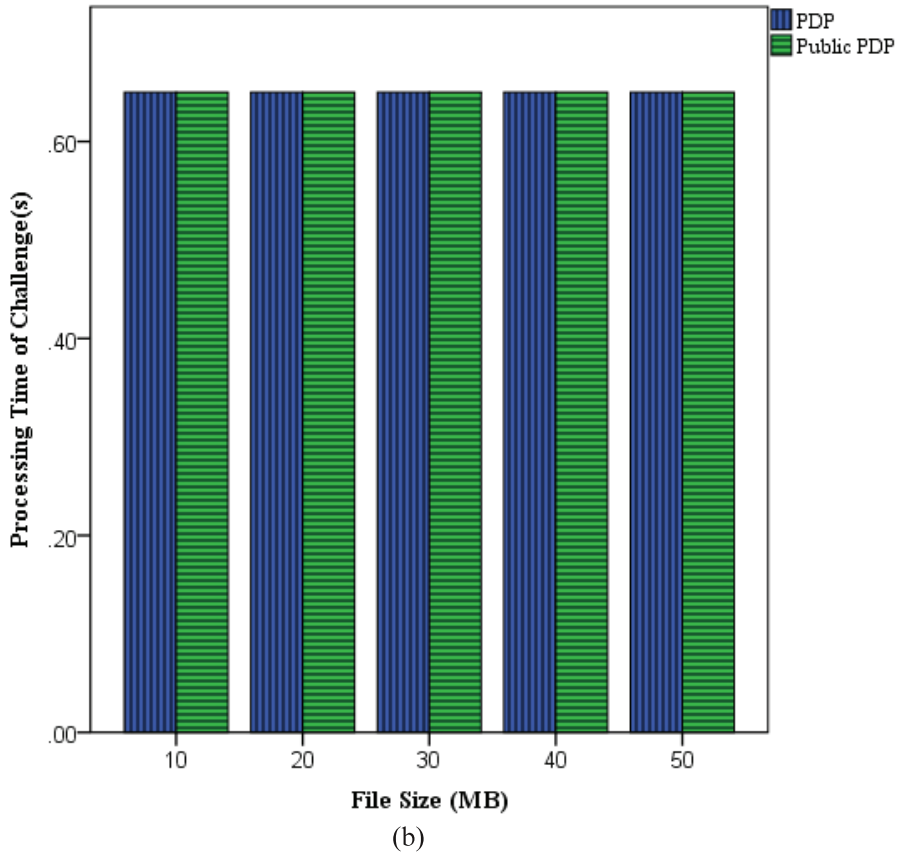
As mentioned in section 2.4.1 of chapter 2, in the challenge step of the traditional RDA methods, the auditor needs to select a number of data blocks randomly on the basis of the rate of data corruption in the server. For example, if the attacker or the untrusted CSP modifies the outsourced data with the rate of 1%, the auditor must randomly choose 230 or 460 blocks of file to detect such misbehavior with probability of 90% or 99% respectively. After selecting the indices of blocks by using pseudo-random permutation technique, the auditor sends the list of indices as challenge to the server. The challenge step in the traditional RDA methods is an identical process and incurs negligible processing time on the auditor. Figure 3.2 shows the computation cost of challenge step in traditional RDA methods.

3.2.2 Processing time of verification step

The processing time of the verification part is the most important cost in RDA schemes. It is because the verification part is carried out several times by the auditor,



(a)



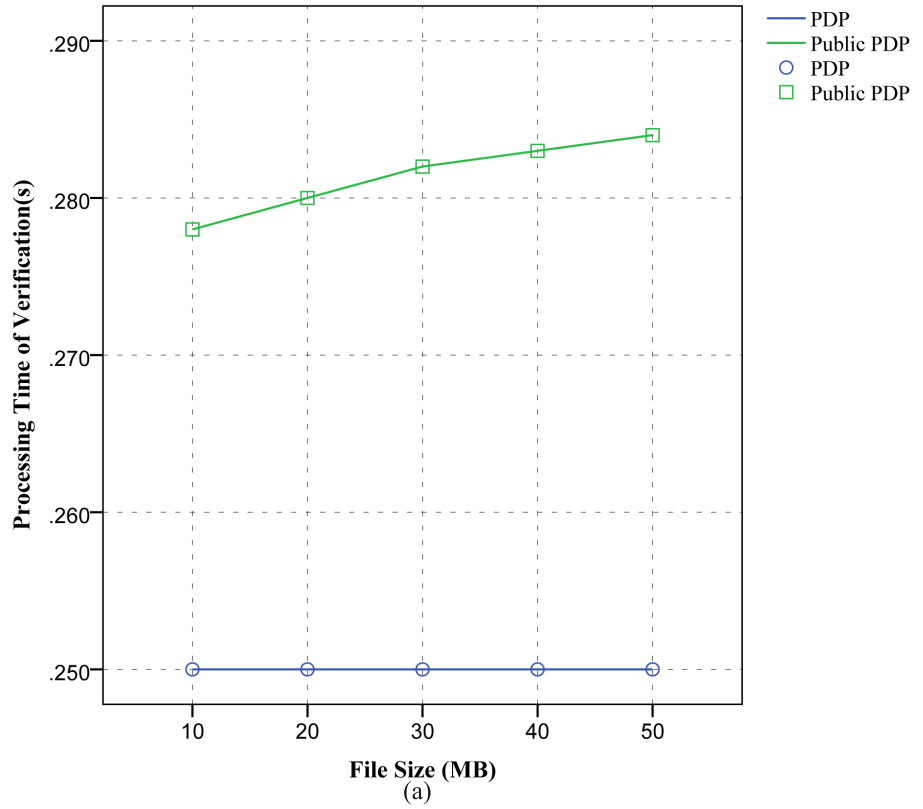
(b)

Figure 3.2: Processing time of challenge step in traditional RDA methods for (a) 90% probability, (b) 99% probability

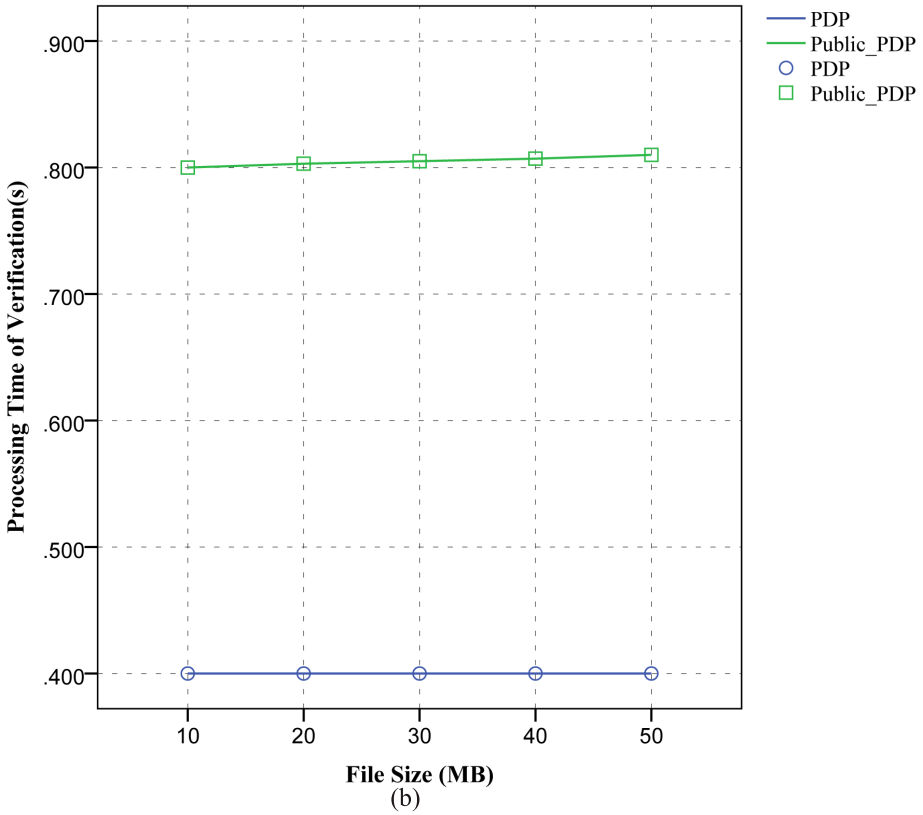
and such cost is directly incurred on the auditor.

When the challenge message is received in the PDP scheme, the CSP generates a proof message including combination (T) of requested tags and the hash of combination of the requested blocks on the basis of challenge message (ρ). The set of (T, ρ) is sent to the auditor as a proof message. Upon receiving the proof, the auditor verify the proof message by using the Homomorphic Verifiable Tag (HVT) cryptosystem. However, verification step in public PDP method incur more computation cost on the auditor due to applying the binary tree for supporting dynamic data update. In other words, when the proof message is received in public PDP method, the auditor needs to re-construct a data structure (that is called Merkle Hash Tree) to authenticate the message, which is a time consuming process and incurs considerable computation cost on the auditor. Finally, the auditor uses the Homomorphic Linear Authenticator (HLA) cryptosystem to verify the proof message. The computation cost of verification step of traditional RDA methods for normal file size in the range of 10 MB to 50 Mb and probability of detection 90% and 99% is illustrated in Figure 3.3. The graph shows that the computation cost of verification step for PDP method is around 0.25 and 0.40 second when the probability of detection is 90% and 99% respectively. Such cost rises dramatically to 0.284 and 0.810 second in the public PDP method for 90% and 99% probability of detection respectively. The main reason of this unwanted additional cost is using the Merkle Hash Tree (MHT) as a data structure.

Analysis of the computation complexity of data integrity indicates that performing remote data auditing methods incur considerable processing time to the client. Decreasing such processing time can play an important role in RDA area because the data owner (auditor) is able to use the smart phones with limited CPU and power resources.



(a)



(b)

Figure 3.3: Processing time of verification step in traditional RDA methods for (a) 90% probability, (b) 99% probability

3.3 Analysis of Communication Cost of Data Update for Normal File Size in Static Integrity-Based Methods

To update a block of the outsourced file, the data owner needs to download a part of data and upload it to the cloud after performing update operations. The transmission cost of dynamic data update indicates the amount of transmitted data between the data owner and the CSP during data update.

During the setup phase of PDP scheme (Ateniese et al., 2011, 2007), the data owner generates a tag for each data block by using the following equation:

$$T_i = (h(v||i).g^{f[i]})^d \bmod N \tag{3.3}$$

Where v and d are the data owner's private key, N and g are the data owner's public key, and $f[i]$ is the i^{th} block of the file. The data owner outsourced the data blocks and tags to the cloud and deletes the local copy of them. Assume that the data owner wants to update the outsourced file by inserting a new block after block j. To achieve this goal, all of the blocks after j must be shifted forward to provide a location for inserting the new block. Since the tag of blocks depends on the location of the blocks, changing the position of the blocks affects the tag of the blocks. Therefore, the data owner must perform the following modifications:

(i) Download the outsourced file completely and retrieve the blocks after j.

(ii) Compute tags for a new block and the blocks after j.

(iii) Outsource the data blocks and tags to the cloud and deleting the local copy of file.

As a result, the communication cost of data update in such method consists of the amount of data blocks that must be downloaded and uploaded for performing update operations. Figure 3.4 illustrates the communication cost of updating a block in PDP

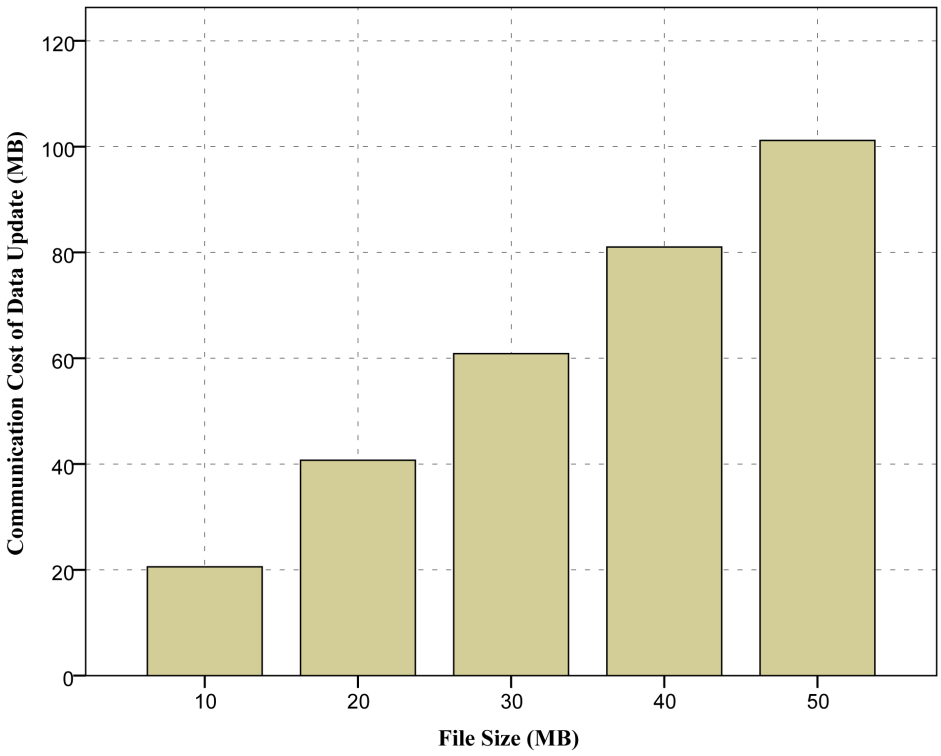


Figure 3.4: Communication Cost of Updating a Block in PDP Method

scheme (Ateniese et al., 2011, 2007) when the size of outsourced is from 10 MB – 50 MB. As it is clear, inserting a single block in the random position of the traditional RDA methods results in considerable communication cost on the auditor. The graph also shows that such cost increases dramatically by increasing the size of file.

3.4 Analysis of Processing Time of data update for Normal File Size in Static Integrity-based Methods

Dynamic data update operations consist of insert, delete, and modify operations. Therefore, the processing time of dynamic data update includes the following costs:

(i) ***Processing time of data insert*** (CT_i): When the data owner wants to update a single block of the outsourced file by inserting a new block, the following process needs to be performed by the data owner: (1) generating the tag of new block, (2) updating the data structure by inserting a new block, (3) checking the integrity of the outsourced data. Hence, the processing time of data insert is computed by Equation

3.4.

$$\sum PT_i = PT_{new-tag} + PT_{structure-update} + \sum PT \tag{3.4}$$

Where $PT_{new-tag}$ indicates the processing time to generate the new tag, $PT_{structure-update}$ indicates processing time to rearrange the data structure on the basis of data operations, and $\sum PT$ is the processing time to check the integrity of the outsourced file.

(ii) ***Processing time of delete*** (PT_d)***:*** To delete a block, the data owner has to: (1) deleting the block and updating the data structure, (2) checking the integrity of the outsourced data. Therefore, the computation cost of delete operation is computed by:

$$\sum PT_d = PT_{structure-update} + \sum PT \tag{3.5}$$

Where $PT_{structure-update}$ is processing time for the data structure rearranging, and $\sum PT$ is the processing time for checking the data integrity.

(iii) ***Processing time of data modification*** (PT_m)***:*** the modify operation includes: (1) generating the tag of the modified block, and (2) checking the integrity of the outsourced data. Consequently, the computation cost of data modification is calculated by:

$$\sum PT_m = PT_{new-tag} + \sum PT \tag{3.6}$$

Where $PT_{new-tag}$ is the processing time to generate the tag of modified block, and $\sum PT$ is the processing time to validate the data integrity.

Therefore, the total processing time of dynamic data update operations $(\sum PT_u)$ including a set of insert, delete, and modify operations is computed by adding the processing time of data insert (PT_i), processing time of delete (PT_d), and processing time of data modification (PT_m), as follows:

$$\sum PT_u = PT_i + PT_d + PT_m \tag{3.7}$$

To show the effect of updating blocks on processing time, a scenario is considered in this research in which a new data block is inserted after a specific block that is selected randomly. This scenario is repeated 10 times to insert 10 data new blocks randomly in each file to demonstrate the effect of frequent data update. This experiment is performed for different normal files (10 MB – 50 MB) in Table 3.1.

Table 3.1: Processing Time of Updating a Block in PDP Method

File Size (MB)	Number of Updates	Processing Time (ms)
10	2	528.477
	4	648.955
	6	730.713
	8	772.354
	10	787.563
20	2	837.422
	4	1261.309
	6	1619.668

Continued on Next Page...

Table 3.1 – Continued

File Size (MB)	Number of Updates	Processing Time (ms)
20	8	1910.137
	10	2155.875
30	2	1176.953
	4	1937.91
	6	2620.918
	8	3223.184
	10	3772.586
40	2	1387.1
	4	2358.972
	6	3257.126
	8	4078.904
	10	4850.359
50	2	1617.568
	4	2822.369
	6	3965.359
	8	5044.309
	10	6081.066

The comparison of processing time for different size of files is also illustrated in Figure 3.6. When the number of update operations are 2, the computation cost is in the

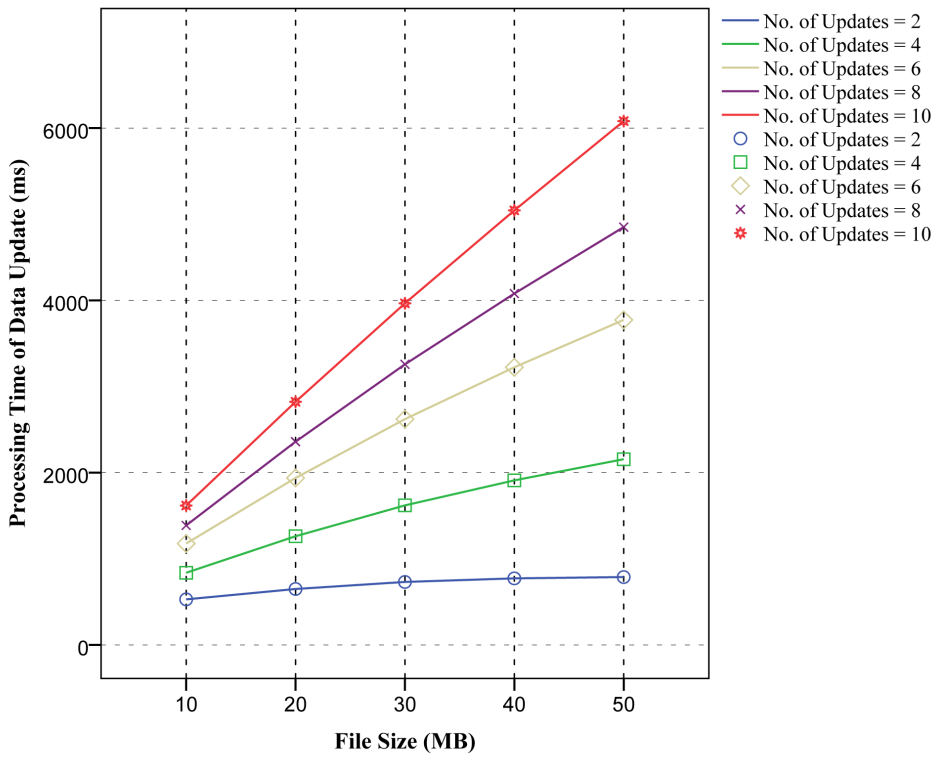


Figure 3.5: Processing Time for Updating Different Blocks in Traditional RDA Methods

range of 528.5 ms – 1617.6 ms for 10 MB – 50MB files respectively. Therefore, it is concluded that updating small number of data blocks in traditional RDA methods incur significant processing time on the auditor.

Analysis the processing time and communication cost of existing RDA schemes shows that performing update operations increase such cost dramatically. Moreover, the size of outsourced block is directly related to the processing time and transmission process of update operations.

3.5 Analysis of Replay attack of data update for Normal File Size in Static Integrity-based Methods

The main disadvantage of most of static integrity-based RDA methods is that the server is able to use the previous version of the data blocks for generating the response message to pass the verifying phase during update operations (that is called replay attack). For example, imagine that the data owner wants to change the i^th block of the outsourced data $(b[i], T[i])$ to $(b'[i], T'[i])$ in PDP method, where $T[i]$ and $T'[i]$ are the tag

of the blocks. When the modified block and corresponding tag $(b'[i], T'[i])$ is received, the cloud server pretends that the requested block and tag are updated accordingly. After that the data owner delete the local copy of the modified block and the corresponding tag. However, there is no way in static integrity-based RDA methods to ensure whether or not the block has been modified. As a result, when the data owner sends a challenge message, the server is able to use the old version of this block to generate a valid proof message for the data owner. It can be conclude that the static integrity-based RDA Method are not able to securely perform data update operations.

3.6 Analysis of Dynamic Data Update Operations for Large Scale File Size in Dynamic Integrity-based Methods

One of the main application of cloud storage is to archive the sensitive data of the ordinary users or organizations. The size of this information can be large and they will be update infrequently. This section analyzes the computation cost of dynamic data update operations on the traditional dynamic integrity-based data auditing methods for the large-scale files that needs to be updated rarely. The size of file is considered between 1 GB and 10 GB, because the maximum file size in most of the cloud storage is less than 10 GB. For example, the maximum size of file in Box is 2 GB to 5 GB (*What's the maximum file size I can upload?*, 2014), and OneDrive is 2 GB to 10 GB (Perez, 2014).

When a part of outsourced file is updated, the data owner needs to ensure that the server is unable to use previous version of updated blocks to generate the response message (Sookhak, Talebian, et al., 2014). One of the effective ways to overcome such an important issue is making use of binary tree data structures, such as Merkle Hash Tree (MHT) (Merkle, 1980). Although MHT data structure decreases the processing time of data update for normal file size and enhance the security of method, such data structures lack in considering the additional cost of updating large-scale files.

To demonstrate the effect of dynamic data update operation on processing time for

large-scale file, a scenario is defined wherein the data owner updates a small number of outsourced blocks (between 2 and 10) by inserting new blocks in random positions or deleting blocks randomly. The size of each block is 8 KB and number of blocks is from 125000 to 1250000. Table 3.2 explains the computation cost of updating a block on applied data structures in the dynamic integrity-based data auditing methods. The attribute file size indicates the size of the outsourced file. The attribute of number of updates shows how many times the outsourced file are updated by inserting a single block in a random position or deleting a random block. Finally, the processing time attribute demonstrates the processing time to perform the update operations.

Table 3.2: Processing Time of Updating a Block in PDP Method

File Size (MB)	Number of Updates	Processing Time (ms)
1	2	1.6122
	4	3.2394
	6	4.8521
	8	6.4803
	10	8.092
2	2	1.6282
	4	3.2404
	6	4.8756
	8	6.4953
	10	8.131

Continued on Next Page...

Table 3.2 – Continued

File Size (MB)	Number of Updates	Processing Time (ms)
	2	1.6304
	4	3.2598
3	6	4.9005
	8	6.5281
	10	8.1603
	2	1.6464
	4	3.2768
4	6	4.9162
	8	6.5476
	10	8.202
	2	1.6469
	4	3.2998
5	6	4.9585
	8	6.5966
	10	8.262
	2	1.6488
	4	3.3051
6	6	4.9611
	8	6.6232
	10	8.279
7	2	1.6642

Continued on Next Page. . .

Table 3.2 – Continued

File Size (MB)	Number of Updates	Processing Time (ms)
7	4	3.3279
	6	4.98
	8	6.6328
	10	8.298
8	2	1.6806
	4	3.3501
	6	4.9937
	8	6.6737
	10	8.326
9	2	1.698
	4	3.3568
	6	5.0145
	8	6.7972
	10	8.3585
10	2	1.7022
	4	3.362
	6	5.0402
	8	6.9102
	10	8.4177

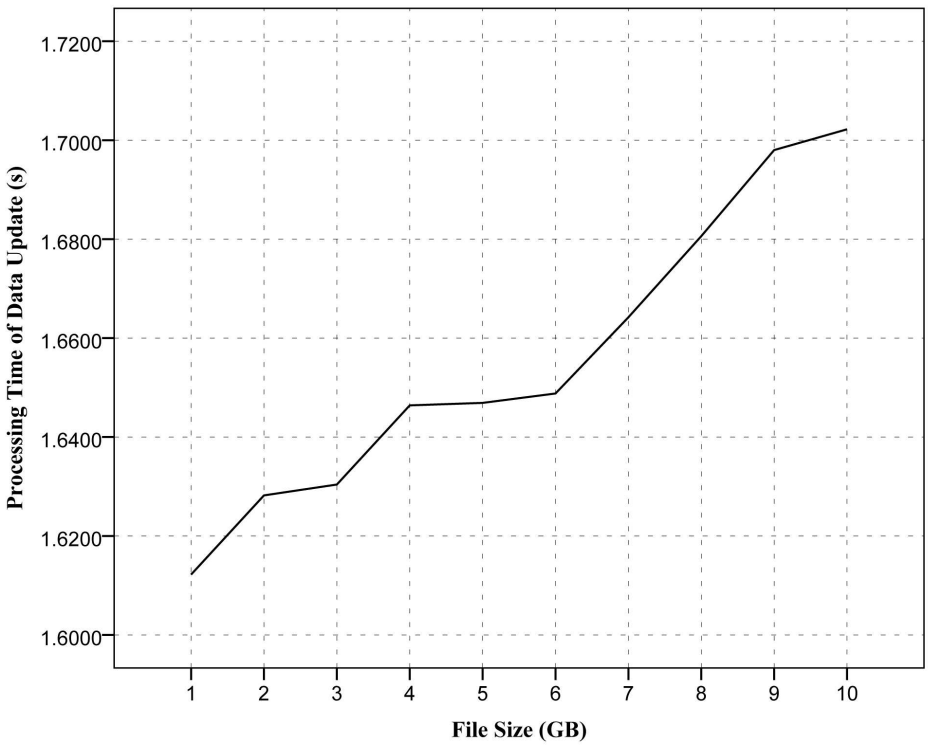


Figure 3.6: Processing time of Data Update in traditional RDA methods when the number of updates is 2

The main drawback of the binary trees is that after inserting a new data block in such data structure (e.g. MHT), the auditor requires to make the tree balance and re-compute the hash of root of the tree through a path from the new block to the root of the tree. Performing this process incur considerable processing time on the auditor.

Figure 3.6 shows the processing time of data update in traditional RDA methods for large-scale files (range in 1 GB to 10 GB) when number of update blocks is 2. It can be seen when the size of file increases from 1 GB to 10 GB, the processing time of data updates is raised from 1.6 s to 1.7 s.

The processing time of data update for large-scale files is illustrated in figure 3.7 when the number of updates is 4. The graph demonstrates that the difference of processing time for a range of 1 GB to 10 GB is around 180 millisecond.

The processing time of data update for 6 and 8 times updates in the large-scale files are displayed in figure 3.8 and figure 3.9. These graphs clearly shows that the difference

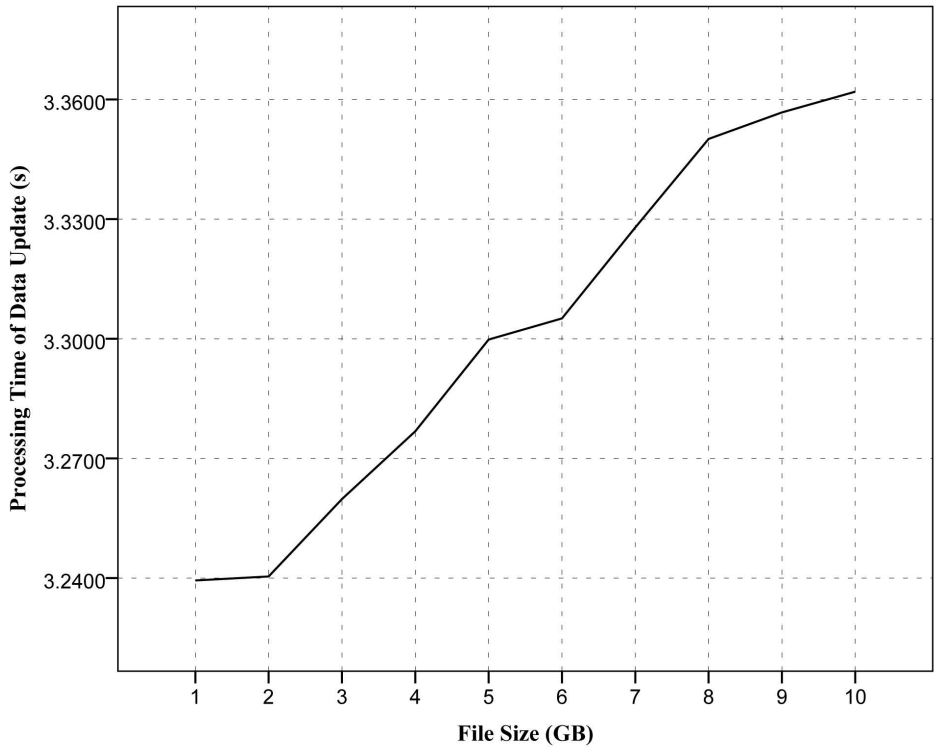


Figure 3.7: Processing time of Data Update in traditional RDA methods when the number of updates is 4

between the processing time of updating a single block in large-scale files is around 200 millisecond to 430 millisecond for 6 and 8 times data update respectively.

The processing time of data update for various range of large-scale files (from 1 GB to 10 GB) is shown in Figure 3.10 when data owner performs data update 10 times. It is clear that increasing the size of file to 10 GB result in additional computation cost on the auditor (around 330 Millisecond).

In summary, re-arranging the data structure in traditional RDA methods (e.g. MHT) caused to noticeable processing time on the auditor even though the number of update is small. In other words, analysis of the defined scenario proves that updating only a small part of the file in large-scale size incurs high processing time on the auditor. Moreover, increasing the size of file has a direct relation with the processing time when the binary tree data structure is used to support dynamic data update in the traditional RDA methods.

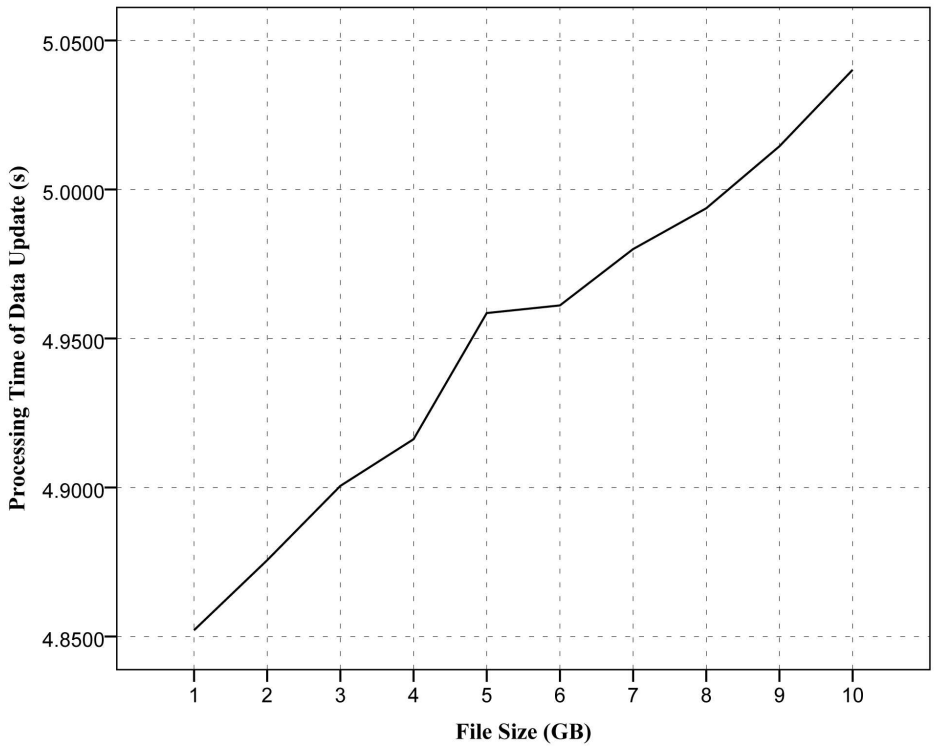


Figure 3.8: Processing time of Data Update in traditional RDA methods when the number of updates is 6

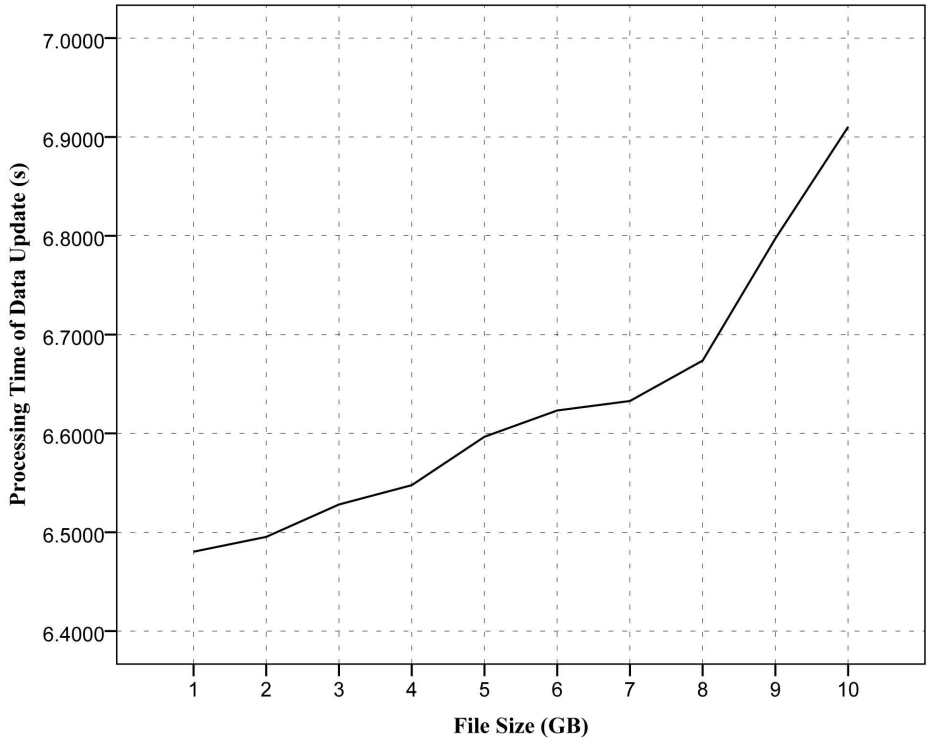


Figure 3.9: Processing time of Data Update in traditional RDA methods when the number of updates is 8

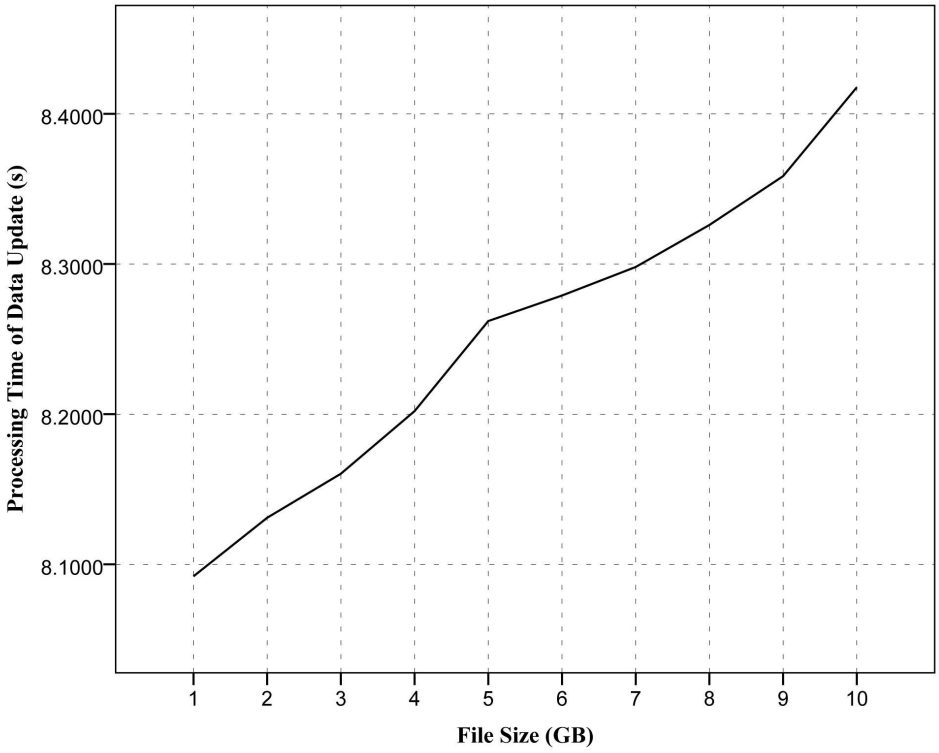


Figure 3.10: Processing time of Data Update in traditional RDA methods when the number of updates is 10

3.7 Analysis of Frequent Data Update on Dynamic RDA Methods for Large-Scale Files

The cloud storage can be used to store the users' data of the large-scale data applications, such as social networks, and bank transactions. These applications allow the users to perform a small but very frequent update (Naone, 2010).

As mentioned earlier in the Section 2.4.1.2 of chapter 2, traditional dynamic integrity-based RDA methods use a specific data structure to perform dynamic data update efficiently, such as MHT (Merkle, 1980). However, to perform frequent data update, these data structures must be re-balanced frequently, which incurs considerable processing time. To evaluate the effect of frequent data update operations on MHT data structure, we consider a scenario wherein the data owner needs to perform 10 different number of update operations (10 – 100) on two different outsourced files 1 GB and 10 GB. The positions of the blocks are selected randomly when the size of each block is 4 KB. The processing time of node re-balancing refers the required time to re-balance the data struc-

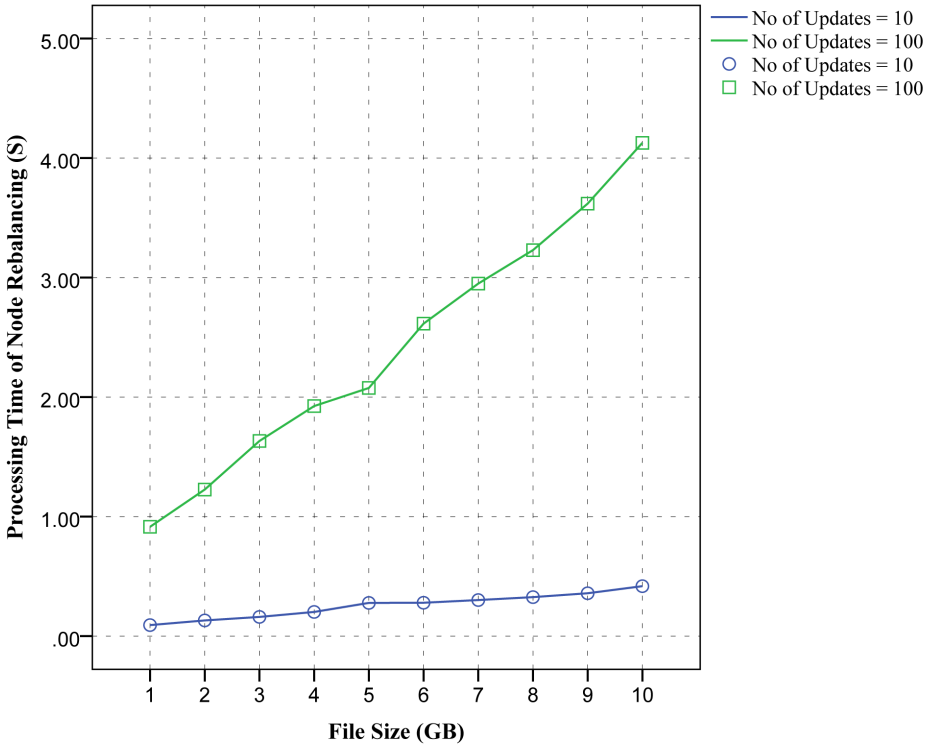


Figure 3.11: Comparison of processing time of node re-balancing for performing 10 and 100 times data updates

ture after inserting a new block in a random position of tree. Figure 3.11 shows the comparison of computation cost of node re-balancing issue in MHT data structure for different large-scale files when the number of data updates is 10 and 100. It can be seen that by increasing the number of data updates in large-scale files, noticeable processing time is incurred on the data owner due to node re-balancing issue.

3.8 Conclusion

Remote data auditing is a useful technique to ensure the integrity of outsourced data in cloud storage systems. However, traditional RDA methods incurs additional processing time on the auditor due to applying homomorphic cryptographic techniques.

Nowadays, the data owner is able to outsource different file size into the cloud storage. Moreover, the stored data in the cloud storage systems can be updated by the data owners frequently or infrequently on the basis of the application. As a result remote

data auditing methods should have capability to support both of frequent and infrequent data update for different size of files effectively. However, current data auditing methods are deficient in deployment of dynamic data update operations when the size of the outsourced file is large. The existing RDA methods focus on performing dynamic data update for normal file size by using different data structures, such as MHT. Therefore, applying such methods for checking the integrity and updating the large-scale outsourced data incurs significant computation cost on the auditor.

The evaluation shows that verifying the integrity of the outsourced data with normal size (10 MB – 50 MB) by traditional remote data auditing methods incurs approximately 400 to 800 millisecond processing time on the auditor. The results demonstrates that updating a single data block infrequently (between 2 and 10 times) when the size of file is 1 GB, incurs 1612 millisecond to 8092 millisecond processing time on the auditor. By increasing the size of the outsourced file to 10 GB, performing infrequent data update (2 – 10 time) incurs more processing time on the auditor (1702 to 8417 millisecond).

The effect of frequent data update on the applied data structure in the traditional RDA methods is also examined by inserting a single block in random position of the MHT data structure. This experiment repeated 10 to 100 times to clarify the additional processing time of node re-balancing issue in traditional methods. The result shows that the processing time of node re-balancing for inserting 10 blocks is from 92 millisecond to 418 millisecond when the size of file is from 1 GB to 10 GB. By increasing the number of update operations to 100, the processing time is raised from 914 millisecond to 4128 millisecond.

Hence, the current remote data auditing methods employ considerable processing time for verifying the integrity and performing update operations in cloud computing. In the next chapter, a new remote data auditing is proposed to fulfill these issues. Moreover, a new data structure is also designed to support dynamic data update efficiently.

CHAPTER 4

DYNAMIC REMOTE DATA AUDITING (DRDA) METHOD

4.1 Introduction

This chapter describes in detail a novel remote data auditing method for addressing the problem of additional processing time in the checking of the integrity of outsourced data in cloud computing. The proposed method uses the characteristics of the algebraic signature technique to effectually verify the correctness of the outsourced data. Moreover, this method has capability to effectively support dynamic data update operations for large-scale files by using a new data structure, namely divide and conquer table. The chapter composes of the following sections.

Section 4.2 explains the proposed the Static Remote Data Auditing (SRDA) method for auditing the outsourced data in the cloud storage by using algebraic signature. Section 4.3 presents the proposed Dynamic Remote Data Auditing (DRDA) method to efficiently support dynamic data operations by introducing a new data structure (Divide and Conquer Table). Finally, the conclusion of this chapter is presented in section 4.4 with highlighting the usefulness of the proposed method.

4.2 Proposed Static Remote Data Auditing Method

We propose a novel dynamic remote data auditing method to verify the correctness of outsourced data in the cloud computing. The proposed method incorporates the features of algebraic properties, which incurs low computation cost on clients and server and makes it suitable for various tasks in large-scale storage systems.

In traditional data auditing methods, the homomorphic cryptosystem is usually used to generate a tag and verify the integrity of the outsourced data remotely that considerable

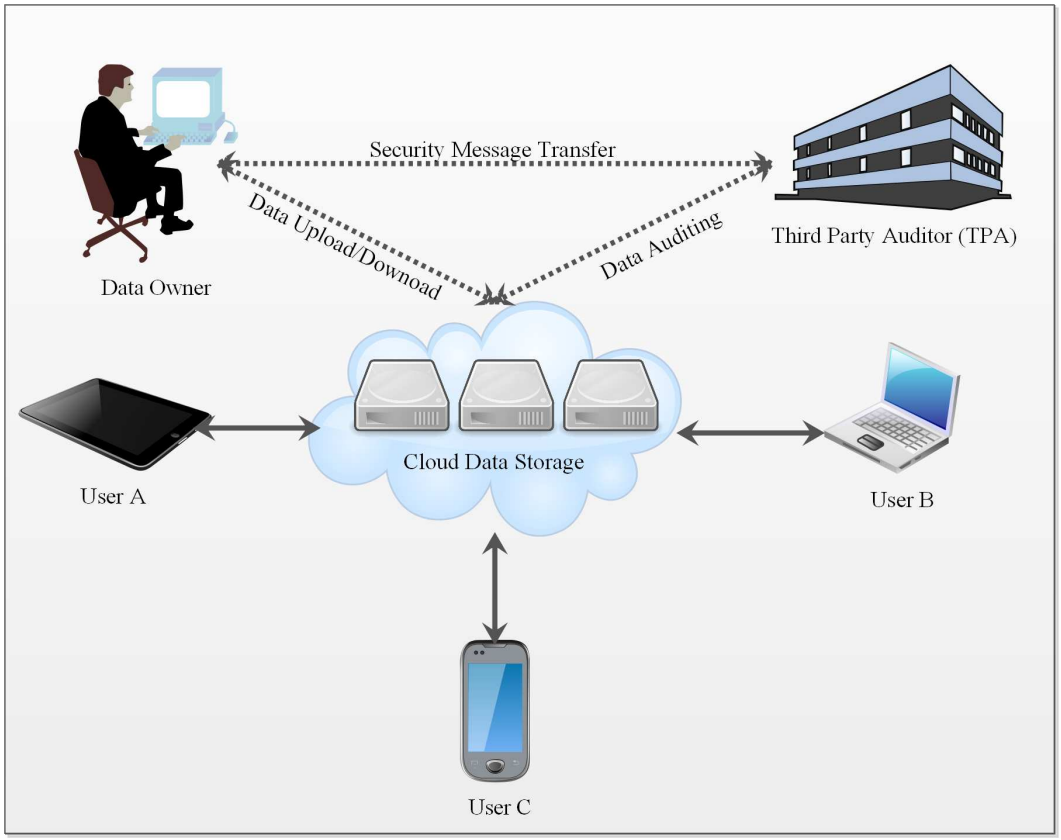


Figure 4.1: Cloud Data Storage Audit Architecture

computation cost on the auditor. To address this issue, Static Remote Data Auditing (SRDA) method focuses on utilizing the algebraic signature for generating the tags and checking the correctness of the data in the cloud storage system.

The architecture of the SRDA method in cloud and mobile cloud computing domain comprises of four main components (Figure 4.1), as follows:

DO who outsources the data in the cloud through a mobile device, laptop or personal computers. DO is able to check the integrity of the data and update the data in the cloud by using insert, append, delete operations.

CSP: is an entity with a large amount of space to store owner's data. The CSP is responsible for generating the response message to prove the possession of the outsourced data.

TPA: Because performing an auditing task introduces performance overheads at the DO

side, the audit task is preferably delegated to an entity called the TPA. The TPA is a trusted entity with expertise and capability to perform the auditing task on behalf of the DO. By employing a TPA, users (especially with limited computing power, storage and connectivity) are alleviated from the burden of expensive auditing tasks. Although the TPA is considered as a trustworthy and reliable entity for data auditing, it may be curious at the same time. Therefore, one of the important issues in the data audit service in the presence of a TPA is to prevent data leakage during auditing and preserve the privacy of data.

User (A/B/C): is an entity (individual or enterprise) who is registered by the DO and granted access to the outsourced data for reading the data.

Remote data auditing services follow response-challenge procedure in which (1) The DO, firstly, pre-processes her file, generates some metadata, and hands over metadata to TPA. At this point, DO is not required to be involved in the audit process anymore, (2) To check correctness of data in the cloud, TPA generates a random challenge message and sends it to the CSP (the DO is also able to generate the challenge message when the TPA is not supported by RDA service), (3) When Cloud storage receives the challenge, computes the corresponding response and sends it to the TPA, and (4) After receiving a response from CSP, the verification is carried out to find out whether or not the CSP has correctly stored the outsourced file. It is important to mention that in order to reduce the overhead of the audit process only a small fraction of whole file is queried.

In the rest of this section, a secure algorithm is presented to remotely audit outsourced data in cloud computing based on the algebraic signature technique. the proposed remote data auditing method consists of the four phases. such as (1) Setup phase, (2) Challenge phase, (3) Response phase, and (4) Verification phase. The following sections explain the aforementioned steps.

4.2.1 Setup Phase

The Data Owner (DO) firstly divides the input file F in to m blocks $F[1], F[2], \ldots, F[m]$. Each data block $F[i]$ is also divided into n sectors indicated as $F[i][1], F[i][2], \ldots, F[i][n]$ by using the data fragment technique. If the number of sectors in a data block is less than n, the DO simply increases the number of sectors to n by appending the number of zero as new sectors to the considered block. Afterward, the DO generates: (1) a secret key by executing the *KeyGen* algorithm, (2) a unique file id (fid) for each file by using *FidGen* algorithm, and (3) a unique tag (T_i) for each block of data file on the basis of the algebraic signature technique by calling the *TagGen* algorithm. Finally, the DO sends the data blocks along with the corresponding tags and file ID $(fid, F[i], T_i, C_i)_{i=1}^{m}$ to the CSP and delete the local copy of the file and tags from the local storage.

$KeyGen(1^k) \rightarrow (pk, sk)$. This probabilistic key generation algorithm takes a security parameter k as an input and produces a pair of public and private key. The private key includes a secret hash key sk_h to compute hash $H(.)$ and a file id secret key sk_{Fid} to sign a message. The public key is also used to verify the signature.

$FidGen(F, sk_{Fid}) \rightarrow (fid)$. This algorithm creates a unique file id (fid) for each file (F). To achieve this goal, the DO (data owner) firstly generates a unique fingerprint $(fprint = Filename||m||n)$ for each file consisting of the file name $(Fname)$, number of blocks (m), and number of sectors (n). After that, the *fid* is generated for each file by concatenating the *fprint* and its signature that is computed under sk_{Fid},as follows:

$$fid = fprint||Sign_{sk_{Fid}}(fprint) \tag{4.1}$$

$TagGen(F, sk_h, sk_{Fid}) \rightarrow (T, C)$. This algorithm is used to compute a single tag for each block based on the algebraic signature of the considering block. The tag of each block (i) consists of two parts, as a block tag (T_i) and tag controller (C_i), that are com-

puted by using the formula:

$$T_i = S_\gamma(f[i]||H_{sk_h}(\tau)) \tag{4.2}$$

$$C_i = n^\gamma \sum_{j=1}^{n} H_{sk_h}(\tau).\gamma^{j-1} \tag{4.3}$$

where, τ indicates the concatenation of *fid* and index of block ($\tau = fid||i$), and γ is a non-zero elements in the Galois field for generating algebraic signature.

4.2.2 Challenge Phase

To verify the integrity of outsourced data, the auditor requires sending a challenge message to the server. The challenge message is composed of a subset of *c* random element of set $[1,m]$ data blocks. The auditor also generates a random coefficient for each block to make sure that server possesses each one of the challenged blocks. The indices and coefficients ($\{cs_i, v_i\}_{i=1}^{c}$) are computed by using pseudo-random permutation (Impagliazzo, Levin, & Luby, 1989) keyed with a fresh randomly-chosen key for each challenge message. It is because of preventing the server from anticipating the block indices. Moreover, the pseudo-random permutation prevents server from keeping a combination of the data blocks instead of the original blocks. Finally, the auditor sends the challenge message ($chal = \{cs_i, v_i\}_{i=1}^{c}$) to the server.

4.2.3 Response Phase

Upon receiving the challenge message ($chal = \{cs_i, v_i\}_{i=1}^{c}$), the server computes the response message on the basis of the challenge message.The response message consists of two parts (μ, σ), as follows.

(i) μ ***part.*** The first part of response message (μ) is computed by linear combining of the selected *c* blocks of the challenge phase. In other words, μ contains *n* sectors

$\mu_1, \mu_2, \cdots \mu_n$ that are computed by making XOR considered sectors of selected c blocks (Equation 4.4).

$$\mu_j = \sum_{i=cs_1}^{cs_c} v_i . f[i][j] \tag{4.4}$$

For example, assume that the auditor selected blocks 2, 5, and 7 as a challenge message and sent them as a challenge message to the server. Therefore, μ involves n sectors $\mu_1, \mu_2, \cdots \mu_n$ of such three blocks that are computed by making XOR the sectors of $2^{th}, 5^{th}$, and 7^{th} data blocks of the outsourced file in the cloud storage, as follows (for simplifying this example, the value of the coefficients(v) are not considered).

$$\mu = \begin{cases} \mu_1 = f[2][1] \oplus f[5][1] \oplus f[7][1] \\ \mu_2 = f[2][2] \oplus f[5][2] \oplus f[7][2] \\ \vdots \\ \mu_n = f[2][n] \oplus f[5][n] \oplus f[7][n] \end{cases} \tag{4.5}$$

Figure 4.2 shows how the sectors of μ $(\mu_1, \mu_2, ..., \mu_n)$ are generated by using n sectors of three blocks $F[2], F[5]$ and $F[7]$.

(ii) σ. ***part*** The second part of the response message (σ) is calculated by aggregating the authenticator tags of the selected blocks in the received challenge by the following Equation.

$$\sigma = \sum_{i=cs_1}^{cs_c} v_i . (T_i \oplus C_i) \tag{4.6}$$

Finally, the server sends the response message (fid, μ, σ) to the verifier.

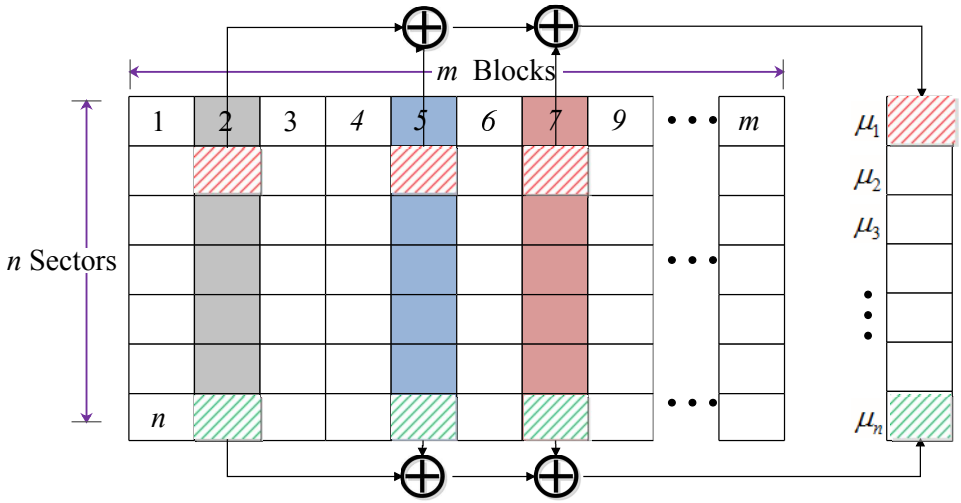


Figure 4.2: Linear combination of requested data blocks as a part of response message (μ).

4.2.4 Verification Phase

When the response message is received, the auditor firstly verifies the signature of the fid under DO's public key (pk) and reject the message if the signature is invalid. Otherwise, the auditor recovers the file name $(fname)$, number of blocks (m), and number of sectors (n). Then, the auditor checks the integrity of the outsourced blocks by:

$$\sigma \stackrel{?}{=} \sum_{j=1}^{n} \mu_j \gamma^{j-1} \tag{4.7}$$

Figure 4.3 shows the process of our data auditing scheme for cloud computing.

4.3 Dynamic Data Operations

Dynamic data update is an important characteristic of auditing schemes that enables the cloud user to update the outsourced data by using insert, delete, modify, and append operations without requiring to download the whole data. However, the server is able to use the previous version of the data blocks for generating the response message to pass the verifying phase during update operations (that is called replay attack). To prevent such attack and support dynamic data update, we propose a data structure that is called Divide and Conquer Table (D&CT). This data structure helps the auditor to store an abstract

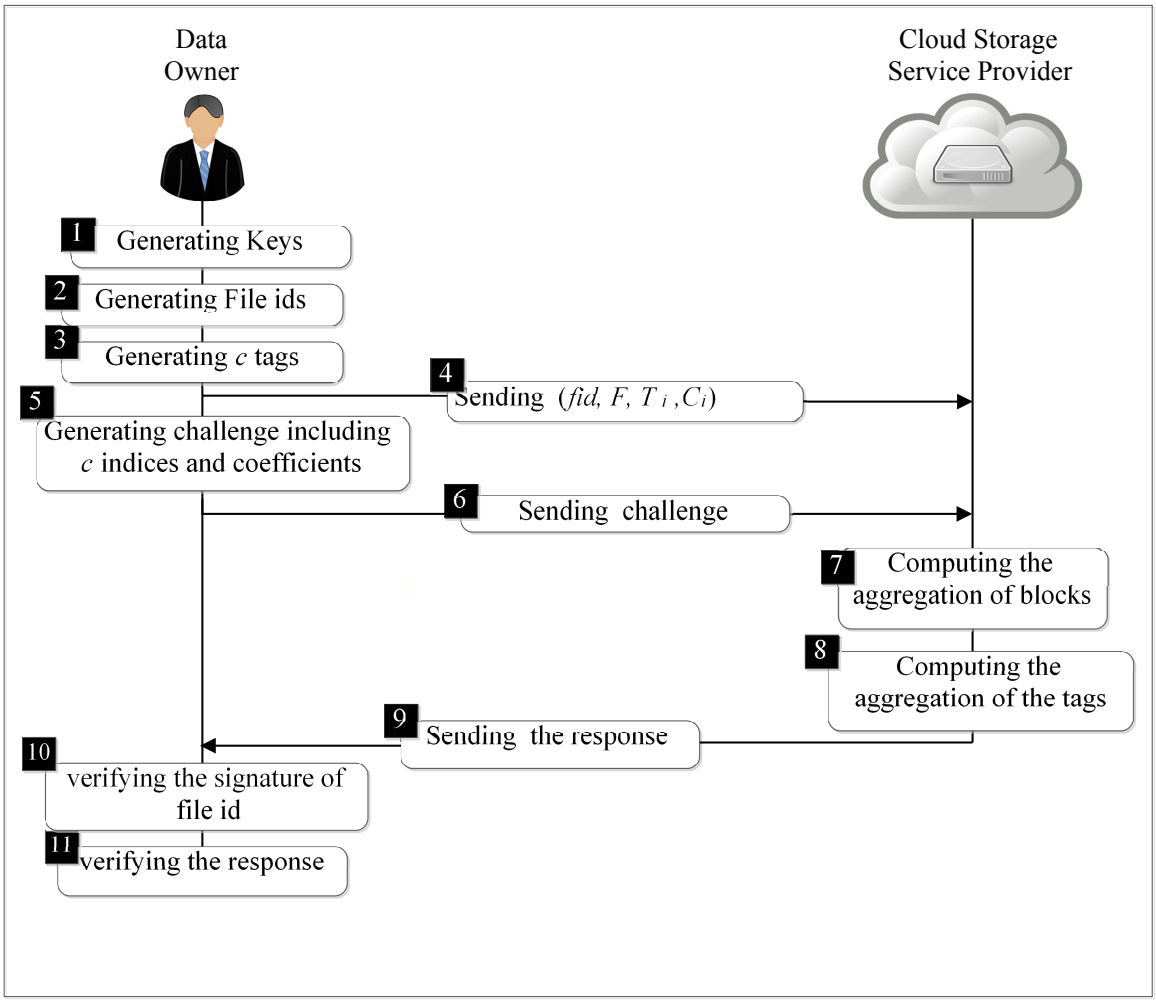


Figure 4.3: Interaction of Component of DRDA Method.

information of data blocks.

The D&CT consists of two components: Logical Index (LI) and Version Number (VN). The LI indicates the original index of data block in the server and the VN indicates the current version of data block on the basis of number of updates. When a data block is updated, the considering VN in the D&CT must be incremented by 1. The index of each block in the D&CT also denotes the physical position of the outsourced data block.

The D&CT data structure must be created by the DO before outsourcing a data block to the cloud. In other words, before generating a tag for each data block during the setup phase, a new entity including $(LI = i, VN = 1)$ is also appended to the D&CT. Moreover, when the DO computes a tag for each data block, the abstract information of data needs to be inserted into the tags by setting $\tau = fid||LI||VN$. It is because to prevent the server

	LI	VN
D&CT [1]	1	1
D&CT [2]	2	1
D&CT [3]	3	1
.		
.		
.		
D&CT [n-1]	n-1	1
D&CT [n]	n	1

Figure 4.4: The structure of Initial D&CT.

to obtain enough information to deceive the auditor by forging the tag from the dynamic operations. Otherwise, the malicious server may forge the tag without having information about the secret hash key when the same τ value is used more than one time. Figure 4.4 shows the details of the proposed data structure (D&CT).

The D&CT must be stored in the local storage of the DO or the auditor who are responsible for managing the D&CT during update operation. Although such data structure empowers our scheme to support dynamic data update operations, managing the D&CT data structure during insert and delete operations imposes high computation overhead on the auditor. For example, to insert a new data block after the i^{th} block, the data owner must shift $n - i$ blocks down. Moreover, the data owner must shift up $n - i$ data blocks for deleting the i^{th} block. Therefore, by increasing the size of file (i.e., large scale files), a huge number of data block must be shifted during insert or delete operations that incurs computation overhead on the client side. Figure 4.5 illustrates the insert and delete operations by using D&CT and their effects of shifting blocks.

To overcome this issue, we reduce the size of the D&CT by dividing it to k data structures in which each of such data structures is able to store $\lceil \frac{n}{k} \rceil$ of the data blocks. As a result, when the DO decides to insert a new block after the i^{th} block, the data owner only needs to shift $\lceil \frac{n}{k} \rceil - i$ data blocks. The experimental results show that the proposed

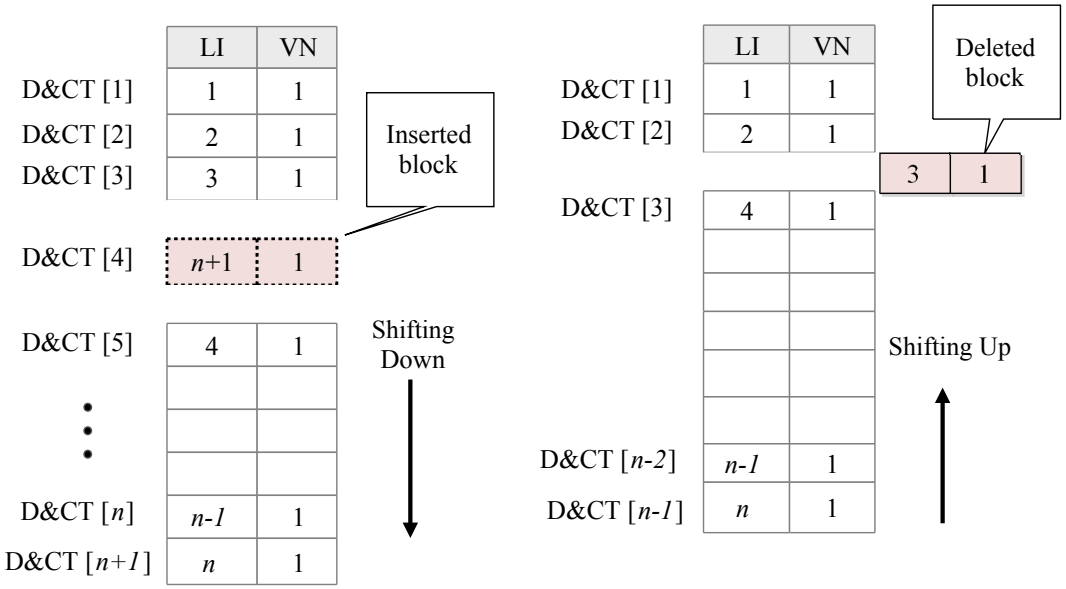


Figure 4.5: Shifting Blocks in Insert and Delete Operations.

data structure is able to support the large scale data efficiently. Each of the D&CTs most also hold the maximum and minimum range of stored data. For example, the range of i^{th} D&CT is from $(i-1)\left[\frac{n}{k}\right]+1$ to $i\left[\frac{n}{k}\right]$. Such ranges need to be modified during insert and delete operations.

The number of divisions (k) is computable by comparing the overhead of these two type of D&CTs during insert and delete operations. As aforementioned previously, the insert or delete operations are carried out in the simple D&CT by shifting the remaining blocks that leads to considerable overhead on the auditor $O(n)$. However, the second data structure only inures $o(\frac{n}{k})$ as a computation overhead on the auditor. It is because the data blocks are stored in k arrays instead of an integrated data structure and the blocks of one D&CT needs to be shifted in each time. Furthermore, in the worst case, our method incurs $O(k)$ to find the location of the block. Therefore, the proposed method is efficient if and only if:

$$k+\frac{n}{k}\leqslant n \Rightarrow k+\frac{n}{k}-n\leqslant 0 \Rightarrow \frac{k^2+n-nk}{k}\leqslant 0 \tag{4.8}$$

Since $k \geqslant 1$, then:

$$k^2 + n - nk \leq 0 \Rightarrow \begin{cases} k_{\min} = \frac{n-\sqrt{n^2-4n}}{2} \\ k_{\max} = \frac{n+\sqrt{n^2-4n}}{2} \end{cases} \tag{4.9}$$

Therefore, the optimal number of divisions is computed by using the following formula:

$$1 - \frac{n}{k^2} = 0 \Rightarrow k^2 = n \Rightarrow k_{opt} = \sqrt{n} \tag{4.10}$$

Table 4.1 shows the minimum, maximum, and optimized number of D&CT tables in the proposed method when the size of outsourced file is between 1 GB and 100 GB and the size of each block is 4 KB.

Table 4.1: Processing Time of Updating a Block in PDP Method

File Size (GB)	**Number of blocks (*n*)**	k_{min}	k_{max}	k_{opt}
1	125000	1.000008	124999	353.55339
2	250000	1.000004	249999	500
3	375000	1.000003	374999	612.37244
4	500000	1.000002	499999	707.10678
5	625000	1.000002	624999	790.56942
6	750000	1.000001	749999	866.0254
7	875000	1.000001	874999	935.41435
8	1000000	1.000001	999999	1000

Continued on Next Page...

Table 4.1 – Continued

File Size (GB)	**Number of blocks (*n*)**	k_{min}	k_{max}	k_{opt}
9	1125000	1.000001	1124999	1060.6602
10	1250000	1.000001	1249999	1118.034
20	2500000	1	2499999	1581.1388
30	3750000	1	3749999	1936.4917
40	5000000	1	4999999	2236.068
50	6250000	1	6249999	2500
60	7500000	1	7499999	2738.6128
70	8750000	1	8749999	2958.0399
80	10000000	1	9999999	3162.2777
90	11250000	1	11249999	3354.102
100	12500000	1	12499999	3535.5339

As mentioned earlier, data owner requires to use the abstract information of each block in such data structure during the tag generation. Since, we divide the D&CT to several data structures, we compel to add the D&CT number to the tag of each block to prevent the server from using the old version of files. As a result, the data owner computes a new τ by:

$$\tau = fid||DN||LI||VN \tag{4.11}$$

Where, fid is the file id, DN shows the block stores in which D&CT, LI indicates the

logical index, and VN is the version number of the block. Finally, the tag of each block is generated by using Equation 2.

In the rest of this section, we discuss how our method performs dynamic data operations, such as modification, insert, delete and append by using D&CT data structure in details.

4.3.1 Data Modification

Supporting data modification is the important requirements of remote data checking techniques in which the Data Owner (DO) has capability to modify a part of the specified block. Suppose that the DO wants to modify the i^{th} block of the file $F(f[i])$ to $f'[i]$. The DO executes the modification algorithm to perform the following modifications:

1. Finding the location of the considered block. The D&CT that holds the required data block can be approximately identified by $k_i = \left\lfloor \frac{i}{\frac{n}{k}} \right\rfloor + 1$ where i indicates the index of block, and $\frac{n}{k}$ is the number of data blocks in each D&CT. Finally, the result is compared by the maximum and minimum ranges of the obtained D&CT (k_i).

2. Increasing the version number of identified block by 1 $(VN' = VN + 1)$.

3. Generating a new block tag for modified data block by:

$$T_i' = S_\gamma(f'[i]\,||H_{sk_h}(fid||DN||LI||VN')) \tag{4.12}$$

$$C_i' = \sum_{j=1}^{n} H_{sk_h}(fid||DN||LI||VN').\gamma^{j-1} \tag{4.13}$$

4. Sending the modification request message to the CSP, which includes $(fid, i, f'[i], T_i', C_i')$ Upon receiving the modification request message, the CSP replaces the block $f[i]$ with $f'[i]$ and update the verion of data block by replacing the tag (T_i, C_i) with

(T_i', C_i'). Figure 4.6 shows that the data owner modify block $f[12]$ when the number of entities in each of table is 5. It is clear that the data owner only needs to increase the NV of this block.

4.3.2 Data Insert

To insert a new data block $(f^*[i+1])$ after the i^{th} block of the file F $(f[i])$, the DO needs to run insert algorithm to perform the following modifications:

1. Finding a D&CT that stores the i^{th} block of the file F and the accurate position of the new block (p) in the found D&CT.

2. Constructing a new row (LI^*, VN^*), inserting it after p^{th} block of the found D&CT, and shifting the subsequent blocks $\lceil \frac{n}{k} \rceil - p$ one position down. The DO also sets the logical index of data block $LI^* = Max(LI) + 1$ and the version number of the block $VN^* = 1$.

3. Increasing upper and lower bounds of subsequent D&CTs by 1. The upper bound of the current D&CT also needs to be increased.

4. Generating a block tag (T^*_{i+1}, C^*_{i+1}) of the new data block by:

$$T^*_{i+1} = S_\gamma(f^*[i+1] || H_{sk_h}(fid||DN||LI^*||VN^*)). \tag{4.14}$$

$$C^*_{i+1} = \sum_{j=1}^{n} H_{sk_h}(fid||DN||LI^*||VN^*).\gamma^{j-1} \tag{4.15}$$

5. Sends the insert request message to the CSP, which includes $(fid, i+1, f^*[i+1], T^*_{i+1}, C^*_{i+1})$. When the CSP receives such the message, the new data block and the considering tag are inserted after position *i* in the file.

Figure 4.6 illustrates how data owner inserts a new data block after 7^{th} block of the input file ($f[7]$). This block is located in 3^{th} row of the second data structure ($D\&CT_2[3]$). Therefore, the DO only needs to shift 3 entities down to insert the new block $(LI^*, VN^*) = (16,1)$. Furthermore, DO must increase upper bound of $D\&CT_2(UB_2 = UB_2 + 1)$, lower bounds of $D\&CT_3(LB_3 = LB_3 + 1)$, and upper bound of $D\&CT_3(UB_3 = UB_3 + 1)$ respectively.

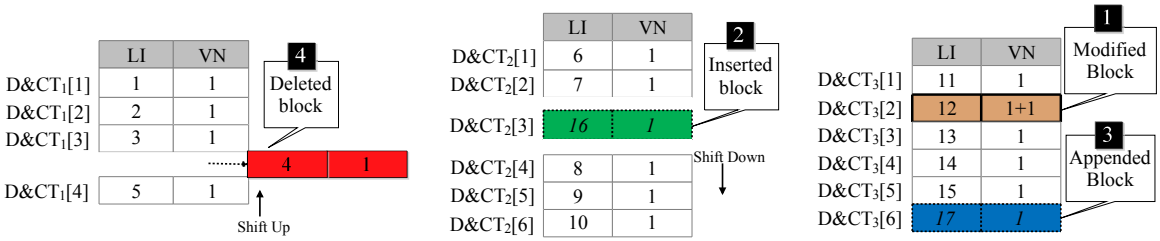
4.3.3 Data Append

The append operation refers to the insertion of a new data block into the end of data blocks. IN this situation, the Do only needs to insert a new row to the end of the last D&CT without having to shift any entities of the D&CTs. For instance, Figure 4.6 shows that to append a new block, the data owner creates a free row for the last table and increases its upper bound $(UB_3 = UB_3 + 1)$.

4.3.4 Data Delete

The delete operation is opposite the insert operation in which the i^{th} block of the file $F(f[i])$ is removed from the D&CTs. To achieve this goal, the DO finds the D&CT that contains the required block ($f[i]$) and the position of the requested block (p) on the basis of the D&CTs ranges. Then, the found block is removed by shifting all of the subsequent blocks $\lceil \frac{n}{k} \rceil - p$ one position up. Moreover, the upper and lower bounds of subsequent D&CTs are decreased along with the up range of current D&CT. Finally, The DO sends a request to delete the i^{th} block of the file to the server.

For example, as it is shown in Figure 4.6, to delete a 4^{th} data block ($f[4]$), the data owner only needs to shift up 1 rows ($f[5]$) and the upper and lower bounds of next tables will be reduced $(LB_2 = LB_2 - 1), (UB_2 = UB_2 - 1), (LB_3 = LB_3 - 1), (UB_3 = UB_3 - 1)$along with the upper bound of the first table $(UB_1 = UB_1 - 1)$.



	LB_1	UB_1	LB_2	UB_2	LB_3	UB_3
Initial	1	5	6	10	11	15
Modify	1	5	6	10	11	15
Insert	1	5	6	11	12	16
Append	1	5	6	11	12	17
Delete	1	4	5	10	11	15

Figure 4.6: Managing modification, Insert, Append, and Delete Operations using D&CT

4.4 Conclusion

We propose a remote data auditing method on the basis of algebraic signature properties, which allows the client to check data possession in cloud storage efficiently while incurring a fewer computational overhead on cloud side and client side compares to Homomorphic cryptosystem. To support dynamic data update operations, such as insert, append, delete, and modify operations, we extend our data auditing method by designing an efficient data structure. The D&CT also helps the auditor to store an abstract information of data blocks to prevent the server from using an old version to pass the verification phase and performing the replay attack.

The D&CT data structure also empowers our method to be applicable for large-scale data storage with least computation cost on client and server. It is because the divisibility as the main feature of our data structure, which reduces the number of block that must be shifted up or down during insert and delete operations.

In the next chapter, we justify the performance of our method, and compare with the stat-of-the-art data auditing methods.

CHAPTER 5

EVALUATION

5.1 Introduction

This chapter evaluates the effectiveness of the proposed DRDA method by using the data collection method. It also explains the tools that was used to test the proposed scheme and the statistical model used to process the collected data. Section 5.2 explains the setup environment, programming tools, and the prerequisites to implement the proposed remote data auditing scheme. It also includes a structure, which described the sampling technique adopted in the data collection based on essential parameters of the proposed scheme. Section 5.3 presents the data collection for evaluating the processing time of proposed RDA method in distinctive phases, such as: setup, challenge, response, and verification. Section 5.4 presents the data collection for evaluating the communication cost of the proposed method in different phases. Section 5.5 presents the data collection for evaluating the processing time of dynamic data update of the DRDA methods for large-scale files (rarely update). The Data collection for evaluating the processing time of frequent file update is presented in section 5.6. Lastly, section 5.7 concludes the chapter.

The proposed auditing method provides a novel solution to verify the integrity of the outsourced data in cloud computing. The public and private data sets were employed to test the DRDA method in different experimental scenarios, and the experimental results are validated by benchmarking in the cloud computing environment.

The experiments include outsourcing and verifying the integrity of the different size of the outsourced files, and updating various number of data blocks of the outsourced files in the cloud. Then, to demonstrates the effectiveness of the proposed method, a variety

of the implementation parameters were employed, such as the processing time and the communication cost of data integrity for the normal files (10 MB – 50 MB) and the large-scale files (1 GB – 10 GB), the processing time of data update for large-scale files (1 GB – 10 GB), and the processing time of frequent data update for large-scale files (1 GB – 10 GB). Each experiment was conducted 20 times and the mean of each experiment (as a processing time or communication cost) is computed and signified by considering 99% confidence interval.

5.2 Evaluation of the Proposed DRDA Method

This section explains the methodology that was adopted to evaluate the proposed method in cloud computing environment. It also presents the experimental setup, the statistical method which illustrates the confidence level of data, along with the data collection structure.

5.2.1 Experimental Result

Experimentation of the proposed remote data auditing scheme was carried out by using a Eucalyptus private Infrastructure as a Service (IaaS) cloud. Eucalyptus stands for "Elastic Utility Computing Architecture for Linking Your Programs to Useful Systems". It is a Linux-based open-source software architecture which can be installed on any Linux operating systems (such as RHEL, Centos, Ubuntu, and Debian) with no need for modification. Eucalyptus was adopted in this thesis due to the following advantages:

- Its compatibility with Amazon AWS APIs [1] which means that we can use Eucalyptus commands to manage Amazon or Eucalyptus instances and move freely between a Eucalyptus private cloud and the Amazon Public cloud making it a hybrid cloud.

[1] http://aws.amazon.com/s3/

- Eucalyptus cloud computing architecture is highly scalable because of its distributed nature and is flexible enough to support businesses of any size.
- It allows you to make your apps in-house on Eucalyptus and then migrate them to the AWS.

Eucalyptus was designed initially at the University of California, Santa Barbara to support high performance computing (HPC) research (Nurmi et al., 2009). The main components the Eucalyptus are summarized as follows.

(i) ***Cloud Controller (CLC)*** is actually the entry-point into the cloud for administrators, managers, developers, and end-users and is accountable for satisfying the request of node managers. CLC is also responsible for making and implementing high level scheduling decisions with the help of cluster controllers.

(ii) ***Cluster Controller (CC)*** generally executes on a computer system that has network connectivity to the systems running Node Controllers (NCs) and to the machine running the CLC. It actually manages a number of VMs and schedules their execution on particular NCs.

(iii) ***Node Controller (NC)*** is executed on every system that is selected for hosting VM instances. It manages the life cycle of instances by making interaction with the OS and the hypervisor running on the same system and the CC.

(iv) ***Storage Controller (SC)*** essentially implements block-accessed network storage such as EBS (Amazon Elastic Block Storage). Subsequently, it has the capability to send disk traffic across the local network to a remote storage site.

(v) ***Walrus*** permits different users to store persistent data. It set access control policies for users to allow certain operations such as delete, create, etc. Its interface is,

however, compatible with Amazon's S3 to store and access both the virtual machine images and user data. It is actually a file-level storage system while essentially represents a block-level storage system.

The experimental setup is implemented using Java and C on a system with an Intel Core i5-2450M CPU at 2.5 GHz, and 4 GB RAM. The Pairing-Based Cryptography (PBC) version 0.5.14 and elliptic curve of the 160-bit group are also used to simulate the existing data auditing schemes (Wang, Wang, Ren, Lou, & Li, 2011; Yang & Jia, 2013). All results are calculated by averaging of 20 trials.

5.2.2 Structure of Data Collection and Analysis

To implement the proposed data auditing scheme, three main parameters were considered: The file size, the length of the signature, and the detection rate (i.e., the probability of misbehavior detection). Figure 5.1 shows the taxonomy of implementation parameters that are used during setup, challenge, response, and verification phases of the data storage auditing method.

The parameter of file size indicates the size of input file, which the DO wants to outsource in the cloud storage. The performance of DRDA method is determined along the footing of two types of files, as (1) normal size in the range of 10 MB to 50 MBs because current cloud storage applications permit the users to outsourced this size of documents, such as Google Drive (*Google Docs, Sheets, and Slides size limits*, 2015). (2) Large-scale file in the range of 1 GB to 100 GB because the maximum file size of the cloud storage is less than 10 GB. For instance, the maximum size of file in Box is 2 GB to 5 GB (*What's the maximum file size I can upload?*, 2014), and OneDrive is 2 GB to 10 GB (*What's the maximum file size I can upload?*, 2014).

The parameter of signature length indicates the number of bit of algebraic signature. The algebraic signature has the capability to generate a short byte string from a large

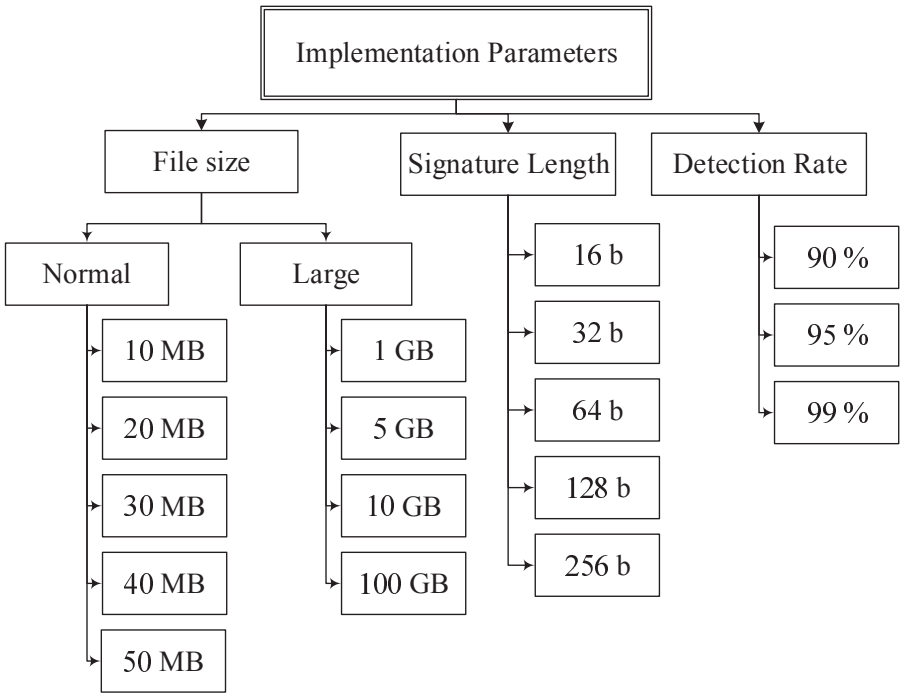


Figure 5.1: Structure of Implementation Parameters

block of the file. However, the byte string must be large enough to decrease the probability of collision and make it extremely unlikely. For Example, a 16 bit signature faces a collision with probability 2^{-16}, while the rate of accidental fit for 256 bit signature is around 2^{-256}. Thus, it is likely impossible to generate a coherent set of signatures without knowing the secrets. The Algebraic signature is computed along the groundwork of defining multiplication by using Galois' theorem GF(2^g) (Savas et al., 2010) as polynomial multiplication modulo a generator polynomial where g can be 16 as half-word or 32 as word. The addition of two polynomials is simply computed by the bitwise XOR of the string. Furthermore, the multiplication by the unknown X is carried out by a left shifting and making XOR with a parameter corresponding to the generator polynomial. As a result, a γ can be identified with the unknown, so that multiplication by γ includes a left shift operation followed by a conditional XOR. Broder (1993) proposed a technique to perform several shift operations at one time, by creating a table consists of a number of decisions that are used as the XOR-operand.

The parameter of detection rate indicates the degree of probability at which the server misbehavior can be detected using the sampling technique. In other words, it indicates the number of blocks of the outsource file that need to be checked by the auditor in order to find the server misbehavior with a specified probability. The probability of detection can be computed using this expression $P_x = 1 - (1 - P_y)^c$ Therefore, if the probability of corrupted blocks $P_y = 0.01$ and the probability of detection $P_x = 90\%$, the auditor needs to check around 230 blocks. To achieve the probability of detection $P_x = 95\%$ and 99% the number of challenge blocks should be around 300 and 460 respectively (more explanation in section 6.3 of chapter 6).

5.2.3 Performance Analysis

There are two important metrics that are applied to examine the efficiency of the remote data checking approaches, such as processing time (computation cost) of data integrity, and communication cost of data integrity.

5.2.3.1 Processing time

As discussed in section 3.2.3.1, remote data auditing methods incur excessive amount of processing time on the auditor and cloud service provider as a prover on the basis of their cryptographic algorithms and applied data structure in the method. Processing time of remote data auditing methods can be evaluated in two situations, such as data integrity, and dynamic data update, as follows:

(1) ***Processing time of data integrity.*** Remote data auditing method involves four phases; setup, challenge, response, and verification. As explained in chapter 4, each phase is responsible to perform an specific task and incurs an amount of computation burden on the auditor or server. As a result, processing time of data integrity of the proposed method includes the following processing times:

(i) ***Processing time of setup phase (PT_s).*** indicates the computational resources that are used by the auditor to divide the file and compute the tags.

(ii) ***Processing time of challenge phase (PT_c).*** is the time that is utilized to generate the challenge.

(iii) ***Processing time of verification phase (PT_v).*** The most important cost in remote data auditing methods is processing time of verification step in which the proof message is checked by the auditor.

(iv) ***Processing time of response phase (PT_r).***) indicates the required time to process and generate the proof message in the response step.

Among these processing times, PT_s, PT_c and PT_v are incurred on the auditor and the cloud server with unlimited computation cost only needs to tolerate the processing time of response step.

(2) ***Processing time of dynamic data update.*** dynamic data auditing methods enable data owners to update the outsourced data by using insert, delete, append, and modify operations. During dynamic data update operations, the data owner needs to accomplish some tasks, such as find the location of requested block, generate new tag, and re-balance the applied data structure on the basis of the update operations. The require time to execute the update operation tasks is called the processing time of dynamic data update (more explanations section 3.2.3.2).

5.2.3.2 Communication Cost

The communication cost shows the amount of data transfer between auditor and server in different phases of auditing scheme. The proposed data auditing methods involves three communication costs, such as communication cost of setup phase, communication cost of challenge phase, and communication cost of response phase, as follows.

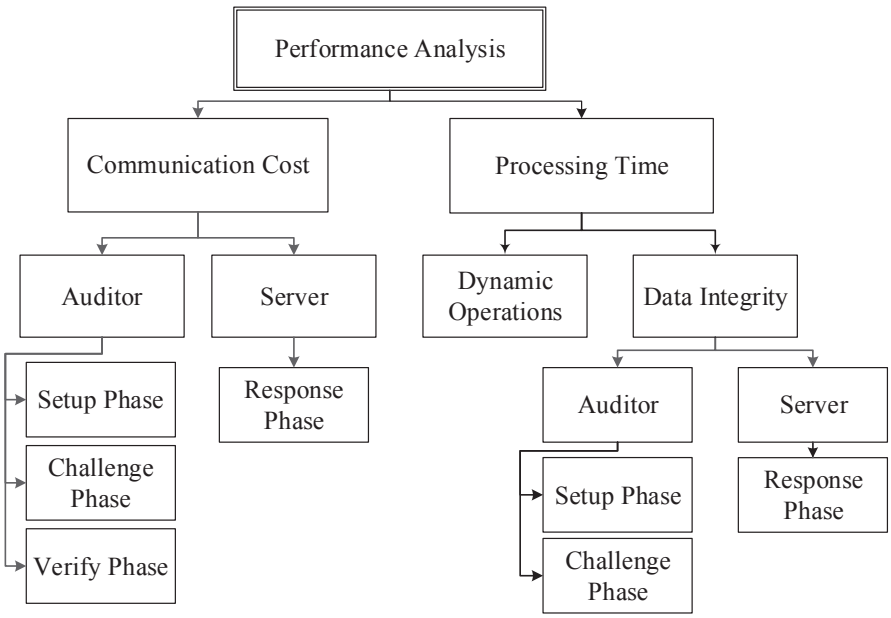


Figure 5.2: Performance analysis parameters

(i) The communication cost of the setup phase indicates the size of data blocks, tags, and other auxiliary files that are sent to server during the setup phase. This communication cost is incurred on the auditor.

(ii) The communication cost of challenge phase includes the size of the challenge message sent to the prover. This communication cost is imposed on the auditor.

(iii) The communication cost of the response phase: the amount of the proof message as a server response that are sent to the verifier is known as communication cost of response phase (imposed on the server).

Figure 5.2 illustrates the performance analysis parameter for the remote data auditing method.

According to the length of the signature and the probability of detection parameters, there are 15 different situations for each of the files that must be considered in evaluation of the proposed method (DRDA). For instance, Figure 5.3 shows the possible number of examination of processing time for outsourcing a 10 MB file to the cloud.

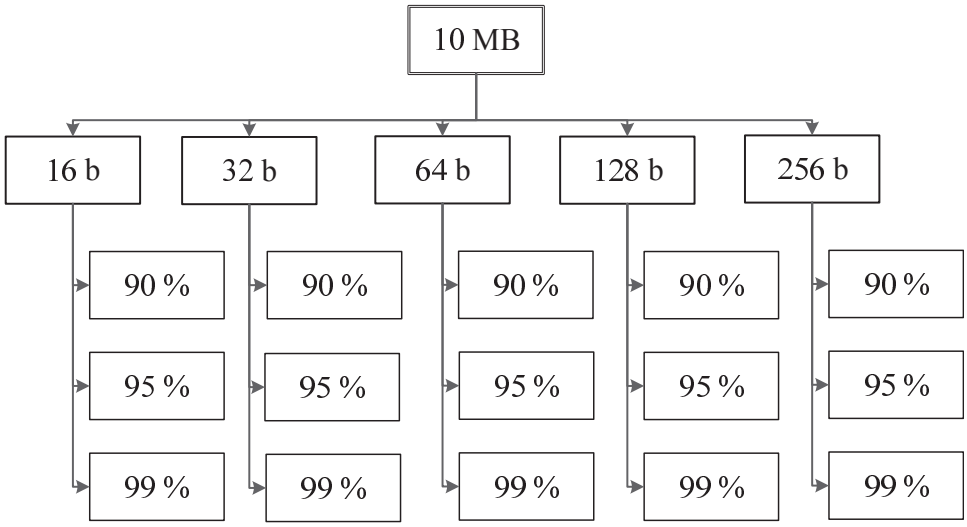


Figure 5.3: The Number of Tests of Processing Time for 10 MB file

5.2.4 Confidence Intervals in Data Gathering

As mentioned earlier in section 5.2, the primary data are gathered by testing the proposed auditing method in different scenarios. Each experiment is conducted 20 times to estimate each parameter based on the sample statistics. A confidence level is a way to show the precision and uncertainty of a selective sampling method, which includes a range of intervals determined from the specified confidence level, a statistic, and a margin of error. The level of confidence is the probability that the parameter is truly captured by the confidence range. The common confidence levels (α) are 90%, 95%, and 99%.

The confidence interval is generally computed by using the Equation 5.1.

$$ConfidenceInterval = SampleMean(CriticalValue)(StandardError) \tag{5.1}$$

The critical value is derived from t-table or z-table on the basis of the level of the confidence (α), number of samples, population standard deviation, and normality of variables. The z-table is used to calculate the confidence interval if: (1) the population standard deviation is known and the number of samples are bigger than 30, or (2) the number of samples are less than 30, population is normally distributed, and the population stan-

dard deviation is known. Equation 5.2 shows how the confidence interval is computed using the z-table.

$$\mu = \bar{X} \pm Z\frac{\sigma}{\sqrt{n}} \tag{5.2}$$

Where μ is the confidence interval, $\bar{X}$ is mean of samples, z is the critical value that extracted from z-table, σ is standard deviation, and n is the number of samples. For example, the critical value (z) for a 95% and 99% confidence interval is 1.96 and 2.58 on the basis of the z-table. Moreover, indicates the standard error (E) that is used for computing the confidence interval. The confidence interval also is in the range of $\bar{X} - ZE$ as a lower bond and $\bar{X} + E$ as an upper bond.

In contrast to z-table, when the population standard deviation is unknown, the number of samples are less than 30, and the population is normal; the confidence interval is computed based on the t-table by:

$$\mu = \bar{X} \pm t_{n-1}\frac{S}{\sqrt{n}} \tag{5.3}$$

Where S is the sample standard deviation, and t_{n-1} extracts from t-table. To eliminate the human error in computing the confidence interval, the IBM SPSS[2] as a well-known statistical software is utilized. The following sections present the data collection in different experiments to evaluate DRDA method.

In the remaining of this chapter, we explain the impractical results of the proposed remote data auditing method in the real environment.

[2]http://www-01.ibm.com/software/my/analytics/spss/products/statistics/

5.3 Data Collection for Processing Time of Data Integrity

This section presents the processing time of the proposed data auditing scheme in the setup, challenge, response, and verification phases respectively.

5.3.1 Processing time of setup phase

In the first step of DRDA method, the data owner needs to divide the file into *m* blocks and each block is divided into *n* sectors. Then, a unique tag is generated for each of the blocks of the input file. Performing this process incurs computational burden on the auditor. However, the setup step only carried out one time for each file. Table 5.1 shows the processing time in the setup phase of the proposed remote data auditing scheme on the client side. The attribute of file size shows the size of an input file that needs to be outsourced to the cloud. The attribute of signature length indicates the length of algebraic signature that is used to generate the tags of the blocks. The attribute of processing time of the setup phase indicates the required time for preprocessing the file.

Table 5.1: Processing Time in the Setup Phase of DRDA method for Normal File Size

File Size (MB)	Signature Length (b)	Processing Time (ms)
10	16	756845
	32	262817
	64	175482
	128	132574
	256	113179
20	16	1540664

Continued on Next Page...

Table 5.1 – Continued

File Size (MB)	Signature Length (b)	Processing Time (ms)
20	32	546013
	64	356409
	128	267600
	256	225753
30	16	2264003
	32	789339
	64	610332
	128	400714
	256	341512
40	16	3054354
	32	1036636
	64	702073
	128	549325
	256	459677
50	16	3771161
	32	1291138
	64	941757
	128	732689
	256	598921

Table 5.2 illustrates the processing time of the setup phase for large-scale data files when the length is 256 b.

Table 5.2: Processing time in the Setup Phase of DRDA method for Large-Scale File Size

File Size (MB)	Signature Length (b)	Processing Time (ms)
1	256	12294697
5	256	61329453
10	256	122968256

5.3.2 Processing time of challenge step

In the second phase, the auditor has to select 230, 300, and 460 blocks randomly and generate 230, 300, and 460 coefficients respectively to check the integrity of the outsourced file with 90%, 95%, and 99% the probability of detection (more explanation about challenge step in section 4.2.2 and explanation about probability of detection in section 6.3). Table 5.3 shows the processing time of the change phase of the proposed data auditing method when the size of files is the normal file size (in the range of 10 MB to 50 MB). The attribute of file size indicates the size of the outsourced file and the attribute of probability of detection indicates the estimation of server misbehavior detection on the basis of the number of blocks in the challenge message. The mean attribute shows the processing time of challenge phase for different file size and probability of detection. The attribute of standard error indicates the standard error of the means of processing time.

The lower bond and upper band attributes also show a lower limitation and an upper limit of the sampling distribution (the mean of processing time) on the basis of t-table or z-table.

Table 5.3: Processing Time of the Challenge Phase for Normal File Size

File Size (MB)	Probability of Detection	Mean of Processing Time (ms)	Standard Error	Lower Bound	Upper Bound
10	90%	11.52	0.60586	9.82545	13.2146
	95%	12.52	1.01673	9.67628	15.3637
	99%	19.64	1.15453	16.4109	22.8691
20	90%	13.12	0.72645	11.0882	15.1518
	95%	15.72	0.91199	13.1692	18.2708
	99%	25.84	0.66	23.994	27.686
30	90%	11.52	0.60586	9.82545	13.2146
	95%	12.52	1.01673	9.67628	15.3637
	99%	19.64	1.15453	16.4109	22.8691
40	90%	12.2	0.73126	10.1079	14.2921
	95%	14.95	0.93323	12.2801	17.6199
	99%	22.3	1.52022	17.9508	26.6492
50	90%	15.9333	0.78962	13.5828	18.2839
	95%	20.4667	0.38873	19.3095	21.6239

Continued on Next Page…

Table 5.3 – Continued

File Size (MB)	Probability of Detection	Mean of Processing Time (ms)	Standard Error	Lower Bound	Upper Bound
50	99%	25	1.18322	21.4778	28.5222

Table 5.4shows the processing time of challenge phase for auditing the large-scale files in different probability of detection.

Table 5.4: Processing Time of the Challenge Phase for Large-Scale File Size

File Size (GB)	Probability of Detection	Mean of Processing Time (ms)	Standard Error	Lower Bound	Upper Bound
1	90%	13.6	0.4	12.4093	14.7907
	95%	15.8	0.2	15.2046	16.3954
	99%	18.7333	0.20625	18.1194	19.3473
5	90%	12.9333	0.26667	12.1395	13.7272
	95%	16.4	0.27255	15.5886	17.2114
	99%	18.6	0.13093	18.2102	18.9898

Continued on Next Page…

Table 5.4 – Continued

File Size (GB)	Probability of Detection	Mean of Processing Time (ms)	Standard Error	Lower Bound	Upper Bound
	90%	12.8	0.27946	11.9681	13.6319
10	95%	15.7333	0.20625	15.1194	16.3473
	99%	17.2	0.31168	16.2722	18.1278

5.3.3 Processing time of Response Phase

The third phase of the DRDA scheme is called response phase in which the server is responsible for generating a linear combination of challenged blocks as a response message. The processing time of the response phase for normal file size is illustrated in Table 5.5. The attribute of file size indicates the size of the outsourced file, which is in the range of 10 MB to 50 MB. The attribute of probability of detection shows the number of selected blocks as a challenge message that is 230 blocks for a 90% probability, 300 blocks for a 95% probability, and 460 blocks for a 99% probability. The mean of processing time attribute presents the average processing time of generating the response message. Ultimately, the standard error, lower bond, and upper bond attributes are computed by using equations 5.2 or 5.3 to show the standard error of the means of processing time, the minimum, and the maximum value of the sampling distribution respectively.

Table 5.5: Processing Time of the Response Phase for Normal File Size

File Size (MB)	Probability of Detection	Mean of Processing Time (ms)	Standard Error	Lower Bound	Upper Bound
10	90%	7607.16	18.6333	7568.7	7645.62
	95%	9910.08	18.0214	9872.89	9947.27
	99%	15224.9	26.2708	15170.7	15279.1
20	90%	7793.92	52.6163	7685.33	7902.51
	95%	10143.4	50.7756	10038.6	10248.2
	99%	15456.8	40.5481	15373.1	15540.4
30	90%	7942.72	72.7489	7792.57	8092.87
	95%	10306.6	32.2553	10240	10373.2
	99%	15768.7	78.2975	15607.1	15930.3
40	90%	7995.85	68.2266	7853.05	8138.65
	95%	10910.4	161.946	10571.4	11249.4
40	99%	15945.4	292.806	15332.5	16558.2
50	90%	7851.73	45.6179	7753.89	7949.57
	95%	10221.9	74.3653	10062.4	10381.4
	99%	15731.7	154.799	15399.7	16063.7

Table 5.6 shows the mean of processing time of generating response message for the large scale files in rage of 1 GB to 10 GB.

Table 5.6: Processing Time of Response Phase for Large-Scale File Size

File Size (GB)	Probability of Detection	Mean of Processing Time (ms)	Standard Error	Lower Bound	Upper Bound
1	90%	7896.47	6.13333	7883.31	7909.62
	95%	10342.6	22.4	10294.6	10390.6
	99%	16091.5	47.8667	15988.9	16194.2
5	90%	7688.2	1.2	7685.63	7690.77
	95%	10674.1	41.4667	10585.2	10763.1
	99%	15842.2	28.8	15780.4	15904
10	90%	7895.8	7.2	7880.36	7911.24
	95%	10380.3	1.06667	10378	10382.6
	99%	15607.5	3.86667	15599.2	15615.8

5.3.4 Processing time of verification step

After getting the response message, the auditor validates the integrity and rightness of the outsourced file in the verification phase. Preforming this process incurs amount of the processing time on the auditor. Table 5.7 presents the processing time of the verifica-

tion phase on the basis of the size of files (normal size in the rage of 10 MB – 50 MB), length of the signature (in the range of 16 b – 256 b), and probability of detection (90%, 95%, and 99%).

Table 5.7: Processing Time of the Verification Phase for Normal File Size

File Size (MB)	Length of Signature (b)	Probability of Detection	Mean of Processing Time (ms)	Standard Error	Lower Bound	Upper Bound
10	16	90%	229.6	0.5099	227.252	231.948
20		90%	230	0.70711	226.744	233.256
30		90%	231.6	0.92736	227.33	235.87
40		90%	232	0.83666	228.148	235.852
50		90%	231	1.18322	225.552	236.448
10		95%	229.6	0.92736	225.33	233.87
20		95%	230.2	0.66332	227.146	233.254
30		95%	230.4	1.43527	223.792	237.008
40		95%	229.8	0.66332	226.746	232.854
50		95%	229.4	0.74833	225.955	232.845
10		99%	243	0.70711	239.744	246.256
20		99%	244.6	1.02956	239.86	249.34
30		99%	244	1.04881	239.171	248.829
40		99%	243.8	0.5831	241.115	246.485

Continued on Next Page. . .

Table 5.7 – Continued

File Size (MB)	Length of Signature (b)	Probability of Detection	Mean of Processing Time (ms)	Standard Error	Lower Bound	Upper Bound
50	16	99%	243	1.41421	236.489	249.511
10		90%	64.6	2.06398	55.0973	74.1027
20		90%	64	1.04881	59.1712	68.8288
30		90%	63.2	1.46287	56.4648	69.9352
40		90%	64	1.87083	55.3865	72.6135
50		90%	63.8	1.06771	58.8842	68.7158
10		95%	69.2	0.86023	65.2394	73.1606
20		95%	68.6	1.8868	59.913	77.287
30	32	95%	69.6	1.43527	62.9919	76.2081
40		95%	68.6	1.02956	63.8598	73.3402
50		95%	69.6	0.67823	66.4774	72.7226
10		99%	70.2	1.82757	61.7857	78.6143
20		99%	71.4	1.63095	63.8909	78.9091
30		99%	70.4	3.20312	55.6525	85.1475
40		99%	70.4	1.02956	65.6598	75.1402
50		99%	71.6	1.07703	66.6412	76.5588
10		90%	31.2	0.66332	28.146	34.254
20	64	90%	31.4	0.8124	27.6596	35.1404
30		90%	31.8	1.11355	26.6731	36.9269

Continued on Next Page. . .

Table 5.7 – Continued

File Size (MB)	Length of Signature (b)	Probability of Detection	Mean of Processing Time (ms)	Standard Error	Lower Bound	Upper Bound
40	64	90%	31.8	0.86023	27.8394	35.7606
50		90%	32	0.70711	28.7444	35.2556
10		95%	34	0.83666	30.1479	37.8521
20		95%	34.6	0.5099	32.2524	36.9476
30		95%	34.8	0.8	31.1167	38.4833
40		95%	34.2	1.15758	28.8704	39.5296
50		95%	34.4	1.02956	29.6598	39.1402
10		99%	34.8	0.86023	30.8394	38.7606
20		99%	35	1.04881	30.1712	39.8288
30		99%	35.4	0.92736	31.1303	39.6697
40		99%	35	1.14018	29.7505	40.2495
50		99%	35	0.70711	31.7444	38.2556
10	128	90%	18.4	0.5099	16.0524	20.7476
20		90%	18.8	0.5831	16.1154	21.4846
30		90%	19	0.83666	15.1479	22.8521
40		90%	18	0.44721	15.941	20.059
50		90%	18.6	0.67823	15.4774	21.7226
10		95%	19.6	0.8124	15.8596	23.3404
20		95%	19.6	1.2083	14.0369	25.1631

Continued on Next Page. . .

Table 5.7 – Continued

File Size (MB)	Length of Signature (b)	Probability of Detection	Mean of Processing Time (ms)	Standard Error	Lower Bound	Upper Bound
30		95%	19.8	0.86023	15.8394	23.7606
40		95%	19.2	0.86023	15.2394	23.1606
50		95%	19.8	0.37417	18.0773	21.5227
10		99%	20.2	0.86023	16.2394	24.1606
20	128	99%	20.6	0.92736	16.3303	24.8697
30		99%	20	0.70711	16.7444	23.2556
40		99%	20.2	0.66332	17.146	23.254
50		99%	20	0.83666	16.1479	23.8521
10		90%	12	0.31623	10.5441	13.4559
20		90%	11.8	0.73485	8.4167	15.1833
30		90%	12	0.70711	8.7444	15.2556
40		90%	11.6	0.92736	7.3303	15.8697
50		90%	11.4	0.92736	7.1303	15.6697
10	256	95%	12.4	0.8124	8.6596	16.1404
20		95%	12.6	0.5099	10.2524	14.9476
30		95%	12.2	0.5831	9.5154	14.8846
40		95%	12.8	0.37417	11.0773	14.5227
50		95%	12.2	0.66332	9.146	15.254
10		99%	13.2	0.37417	11.4773	14.9227

Continued on Next Page. . .

Table 5.7 – Continued

File Size (MB)	Length of Signature (b)	Probability of Detection	Mean of Processing Time (ms)	Standard Error	Lower Bound	Upper Bound
20	256	99%	13.4	0.5099	11.0524	15.7476
30		99%	13.6	0.5099	11.2524	15.9476
40		99%	13.2	0.96954	8.7362	17.6638
50		99%	13	0.70711	9.7444	16.2556

The processing time of verification phase for large-scale file size (in range of 1 GB – 10 GB) is shown in Table 5.8.

Table 5.8: Processing Time of the Verification Phase for Large Scale File Size

File Size (GB)	Length of Signature (b)	Probability of Detection	Mean of Processing Time (ms)	Standard Error	Lower Bound	Upper Bound
1	256	90%	12.6	0.67823	9.4774	15.7226
5		90%	12.2	0.37417	10.4773	13.9227
10		90%	12.8	0.5831	10.1154	15.4846

Continued on Next Page…

Table 5.8 – Continued

File Size (GB)	Length of Signature (b)	Probability of Detection	Mean of Processing Time (ms)	Standard Error	Lower Bound	Upper Bound
1	256	95%	13	1.04881	8.1712	17.8288
5		95%	13.6	1.43527	6.9919	20.2081
10		95%	13.2	0.8	9.5167	16.8833
1		99%	16.6	0.74833	13.1546	20.0454
5		99%	16.6	1.1225	11.4319	21.7681
10		99%	15.8	0.66332	12.746	18.854

5.4 Data Collection for Communication Cost of Data Integrity

Communication cost is an important standard that is employed to measure the execution of the remote data auditing methods. In the remaining of this section, the data collection of communication cost of setup phase, challenge phase, and response phase are studied respectively.

5.4.1 Communication Cost of Setup Phase

The amount of transferred data from the data owner to the server during the setup phase is known as an actual communication cost of setup phase. The transferred data in the setup step consist of the blocks of the file, tags, and some auxiliaries. In this study, to highlight the overload of setup phase, the size of input file is excluded from the communication cost. Therefore, the imposed communication cost on the auditor is

computed by:

$$ImposedCommunicationCost = actualCommucnicationCost - FileSize \quad (5.4)$$

Table 5.9 presents the communication cost of setup phase for outsourcing the normal file size. It can be seen that when the size of file is 10 MB and the size of signature is 16 b, the actual communication cost of setup phase is around 10.00976563 MB and the imposed communication cost is 0.00976563 MB.

Table 5.9: Communication cost of Setup phase for Normal File Size

File Size (MB)	Signature Length (b)	Imposed Communication Cost (MB)
10		0.00991
20		0.01967
30	16	0.02944
40		0.0392
50		0.04897
10		0.01967
20		0.0392
30	32	0.05873
40		0.07827
50		0.0978
10	64	0.0392

Continued on Next Page...

Table 5.9 – Continued

File Size (MB)	Signature Length (b)	Imposed Communication Cost (MB)
20	64	0.07827
30		0.11733
40		0.15639
50		0.19545
10	128	0.07827
20		0.15639
30		0.23452
40		0.31264
50		0.39077
10	256	0.15639
20		0.31264
30		0.46889
40		0.62514
50		0.78139

Table 5.10 shows the communication of the setup phase for uploading large-scale file (in range of 1 GB to 10 GB) into the cloud storage.

Table 5.10: Communication cost of Setup phase for Large-Scale File Size

File Size (GB)	Signature Length (b)	Imposed Communication Cost (GB)
1	16	0.00112
2		0.00209
3		0.00307
4		0.00405
5		0.00502
6		0.006
7		0.00698
8		0.00795
9		0.00893
10		0.00991
1	32	0.00209
2		0.00405
3		0.006
4		0.00795
5		0.00991
6		0.01186
7		0.01381
8		0.01577
9		0.01772

Continued on Next Page. . .

Table 5.10 – Continued

File Size (GB)	Signature Length (b)	Imposed Communication Cost (GB)
10	32	0.01967
1	64	0.00405
2		0.00795
3		0.01186
4		0.01577
5		0.01967
6		0.02358
7		0.02748
8		0.03139
9		0.0353
10		0.0392
1	128	0.00795
2		0.01577
3		0.02358
4		0.03139
5		0.0392
6		0.04702
7		0.05483
8		0.06264
9		0.07045
10		0.07827

Continued on Next Page. . .

Table 5.10 – Continued

File Size (GB)	Signature Length (b)	Imposed Communication Cost (GB)
1	256	0.01577
2		0.03139
3		0.04702
4		0.06264
5		0.07827
6		0.09389
7		0.10952
8		0.12514
9		0.14077
10		0.15639

It is important to mention that similar to the processing time of setup step, the effect of communication cost of setup phase has not been considered by the researchers. This is because the setup step is carried out one time for each file.

5.4.2 Communication Cost of Challenge Phase

To verify the integrity of the outsourced data blocks, the auditor must select a number of blocks randomly as a challenge message on the basis of probability of detection rate. For example, in the proposed method, the auditor has to randomly select the index of 230 blocks, 300 blocks, or 460 for achieving 90%, 95%, or 99% probability of misbehavior detection. The amount of transferred data in this step is known as a communication cost

of challenge step. Table 5.11 shows the computation cost of transferring the challenge messages to the server for normal file size on the basis of length of the signature and probability of detection attributes.

Table 5.11: Communication Cost of Challenge phase for Normal File Size

File Size (MB)	**Signature Length (b)**	**Probability of Detection**	**Communication Cost (KB)**
10-50	16-256	90%	0.89844
10-50	16-256	95%	1.17188
10-50	16-256	99%	1.79688

The communication cost of challenge phase for verifying the integrity of large-scale file size in range of 1 GB to 10 GB is also presented in Table 5.12.

Table 5.12: Communication Cost of Challenge phase for Large Scale File Size

File Size (GB)	Signature Length (b)	Probability of Detection	Communication Cost (KB)
1-10	16-256	90%	0.89844
1-10	16-256	95%	1.17188
1-10	16-256	99%	1.79688

5.4.3 Communication Cost of Response Phase

After receiving the challenge message and generating the response message, the prover requires to transfer the response message to the auditor. The amount of the transferred message to the auditor as a response message is referred to the communication cost of the response phase. Table 5.13 shows the communication cost with subject to length of signature and probability of detection attributes for normal file size (10 MB – 50 MB).

Table 5.13: Communication Cost of Response phase for Normal File Size

File Size (MB)	Signature Length (b)	Probability of Detection	Communication Cost (KB)
10-50	16	90%-99%	4.00391
10-50	32	90%-99%	4.00781
10-50	64	90%-99%	4.01172
10-50	128	90%-99%	4.01563
10-50	256	90%-99%	4.03125

The communication cost of the response message or large-scale files in range of 1 GB to 10 GB is also illustrated in Table 5.14.

Table 5.14: Communication Cost of Response phase for Large-Scale File Size

File Size (GB)	Signature Length (b)	Probability of Detection	Communication Cost (KB)
1-10	16	90%-99%	4.00390625
1-10	32	90%-99%	4.0078125

Continued on Next Page. . .

Table 5.14 – Continued

File Size (GB)	Signature Length (b)	Probability of Detection	Communication Cost (KB)
1-10	64	90%-99%	4.01171875
1-10	128	90%-99%	4.015625
1-10	256	90%-99%	4.03125

5.5 Data Collection for Processing Time of Dynamic Data Update Operations for Large Scale File Size

As aforementioned in section 3.2.3 of chapter 3, the processing time of dynamic data updates indicates the required time to perform update operations such as insert, delete, and insert a block. To demonstrate the effect of dynamic updates on the DRDA method, a scenario defined in which a data owner insert or delete a random data block. This scenario can be repeated several times to show the effect of number of update operations on large scale files in the DRDA method. Table 5.15 shows the processing time of dynamic data update for large-scale files.

Table 5.15: Processing Time of Data Update for Large-Scale Files

File Size (GB)	Signature Length (b)	Number of updates	Processing Time of Data Update (s)
1-10		2	0.032
1-10		4	0.064
1-10	256	6	0.096
1-10		8	0.144
1-10		10	0.176

5.6 Data Collection for Processing Time of Frequent Data Update for Large-Scale Files

Performing data update operations (including insert, delete, and append) on the large-scale files can incur noticeable processing time on the auditor. This is because the auditor need to re-balance the applied data structure in the data auditing methods. Table 5.16 shows the processing time of node re-balancing for frequent updates of large-scale file in the proposed method. The attribute of processing time of node re-balancing indicates the require time for re-structuring and re-arranging the nodes in the data structure of the method. The attribute of the number of update indicates how many data operations are carried out by the data owner. The attribute of file size indicates the size of the outsourced file. The attribute of signature length indicates the size of algebraic signature that is used in the DRDA method.

Table 5.16: Processing Time of Data Update for Large-Scale Files

File Size (GB)	Signature Length (b)	Number of updates	Processing Time of Node Re-balancing (ms)
1	256	10	0.0001
2			0.0001
3			0.0003
4			0.0005
5			0.0007
6			0.0009
7			0.001
8			0.0012
9			0.0014
10			0.0015
1	256	100	0.088
2			0.09
3			0.094
4			0.099
5			0.11
6			0.115
7			0.119
8			0.124

Continued on Next Page. . .

Table 5.16 – Continued

File Size (GB)	Signature Length (b)	Number of updates	Processing Time of Node Re-balancing (ms)
9	256	100	0.124
10			0.125

5.7 Conclusion

This chapter presented the data collection method for examining the proposed remote data auditing scheme based on two essential parameters: processing time and communication cost.

The DRDA scheme is tested on the real environment, and the benchmarking is used to evaluate such a scheme. The data collection is performed by sampling the critical evaluation parameters with in two different groups of files, such as (1) Normal file in the range of 10 MB – 50 MB, and (2) Large-scale file in the range of 1 GB – 10 GB. The data of each experiment is collected in the sample space of 20 values. The mean value for a sample set of data in each of the experiments is computed based on the 99% confidence interval.

As a conclusion, the DRDA scheme successfully leveraged the algebraic signature properties to audit the integrity of the outsourced data in cloud computing. The evaluation of the DRDA scheme in the real environment shows the viability of the proposed method for performing data auditing remotely in different scenarios for various normal and large-scale files.

The next chapter analyzes the presented results of this chapter to indicate the usefulness of the proposed DRDA scheme.

CHAPTER 6

RESULTS AND DISCUSSIONS

6.1 Introduction

This chapter presents the result of the experimental research of the proposed remote data auditing scheme by using Eucalyptus as a private cloud. Moreover, the discussion section goes one step further in looking over the results by comparing the result analysis to show the validity of the proposed scheme. The conducted experimental research is expected to fulfill the following objectives:

(i) To demonstrate the superiority of the proposed data auditing scheme to preserve the integrity of outsourced data in the cloud computing in terms of processing time and communication cost.

(ii) To find out the length of signature, which results in minimum processing time on the auditor.

(iii) To show the dynamicity feature of the proposed method by presenting the D&CT data structure and identify the feasibility of this scheme in facing the large-scale file.

(iv) To demonstrate that the proposed method is viable for normal and large scale data volume by evaluating the performance of the method in different scenarios.

The rest of this section is organized as follows. Section 6.2 proves the security of the proposed scheme, and section 6.3 analyses the security strength of the DRDA method based on sampling strategy. Section 6.4 presents the empirical result of the processing time of the DRDA method in different phases, such as: setup, challenge, response, and

verification. The experimental result of the communication cost of the proposed scheme is also explained in section 6.5. Section 6.6 presents the performance analysis of D&CT data structure. Section 6.7 shows the effectiveness and validity of the proposed scheme by comparing the result with state-of-the-art methods. Finally, section 6.8 presents summary and conclusion of this chapter.

6.2 Security Analysis

This section presents the evaluation of the security of the proposed remote data auditing construction in terms of security and correctness.

The DRDA scheme relies on the algebraic signature that generates a small entity as a signature for each block and is able to show any modifications in the original block. The algebraic signature also has the capability to verify a large amount of stored data on the distributed storage systems with minimum processing time and communication overhead (Schwarz & Miller, 2006). As a result, the algebraic signature can be used for verifying the correctness of outsourced data specially by using the resource restricted devices. On the other hand, probability of collision in the algebraic signature is negligible (Litwin & Schwarz, 2004). For example, if the length of signature is 64 bits, the probability of collision is very small (2^{-64}). In the following, two main properties of the algebraic signature are listed that are used to proof the correctness of the proposed method.

Proposition 1. The algebraic signature of concatenation of two files F with length l and G is computed by $S_\gamma(F||G) = S_\gamma(F) \oplus l^\gamma S_\gamma(G)$ (Litwin & Schwarz, 2004).

Proposition 2. The summation of algebraic signature of m blocks of a file F is equal to the algebraic signature of summation of the blocks.

$$S_\gamma(f[1]) + S_\gamma(f[2]) + \ldots + S_\gamma(f[m]) = S_\gamma(f[1] + f[2] + \ldots + f[m]) \qquad (6.1)$$

Proof. Assume that the File F is divided into m blocks and each of the blocks consists of n sectors. Then:

$$\begin{aligned} S_\gamma(f[1])+S_\gamma(f[2])+...+S_\gamma(f[m]) &= \sum_{j=1}^{n} f[1][j].\gamma^{j-1} + \sum_{j=1}^{n} f[2][j].\gamma^{j-1} + ... + \sum_{j=1}^{n} f[m][j].\gamma^{j-1} \\ &= \sum_{j=1}^{n} (f[1][j] + f[2][j] + ... + f[m][j]).\gamma^{j-1} \\ &= \sum_{j=1}^{n} (\sum_{i=1}^{m} f[i]).\gamma^{j-1} \\ &= S_\gamma(f[1] + f[2] + \ldots + f[m]) \end{aligned}$$

As mentioned earlier in section 4.2 of chapter 4, after generating the public and private keys, the DO (data owner) generates a unique fid for the input file. Then, the DO calculates a unique tag (T_i, C_i) for each block of the file by using equation 4.2, and 4.3. Finally, the DO outsources the file *id*, data blocks, and tags to the cloud. Since the algebraic signature of concatenation of two files F with length l and G is computed by $S_\gamma(f[i]||f[j]) = S_\gamma(f[i]) \oplus l^\gamma S_\gamma(f[j])$ (proposition 1), It is concluded that:

$$T_i = \sum_{j=1}^{n} F[i][j].\gamma^{j-1} \oplus n^\gamma \sum_{j=1}^{n} H_{sk_h}(\tau).\gamma^{j-1}$$

After receiving the challenge message, the CSP who replies decently to a query $(\{cs_i, v_i\}_{i=1}^{c})$, generates a proof message including fid, μ and σ by:

$$\mu_j = \sum_{i=cs_1}^{cs_c} v_i.f[i][j]$$

$$\sigma = \sum_{i=cs_1}^{cs_c} v_i.(T_i \oplus C_i)$$

By extending σ on the basis of the properties of algebraic signature, it can be seen that:

$$\sigma = \sum_{i=cs_1}^{cs_c} v_i.((\sum_{j=1}^{n} f[i][j].\gamma^{j-1} \oplus n^{\gamma} \sum_{j=1}^{n} H_{sk_h}(\tau).\gamma^{j-1}) \oplus n^{\gamma} \sum_{j=1}^{n} H_{sk_h}(\tau).\gamma^{j-1})$$
$$= \sum_{i=cs_1}^{cs_c} v_i. \sum_{j=1}^{n} f[i][j].\gamma^{j-1}$$
$$= \sum_{j=1}^{n} \sum_{i=cs_1}^{cs_c} v_i.f[i][j].\gamma^{j-1}$$
$$= \sum_{j=1}^{n} \mu_j \gamma^{j-1}$$

As a result, the correctness of the verification phase of the proposed scheme is proved $(\sigma \stackrel{?}{=} \sum_{j=1}^{n} \mu_j.\gamma^{j-1})$.

6.3 Security Strength

The proposed remote data checking scheme is constructed on the basis of a random sampling strategy to reduce the workload on the server. In the sampling technique, the auditor requires selecting a random number of blocks (c) out of m blocks of the file (F) to perform batch processing. We analyze the probability of misbehavior detection of our scheme based on the block sampling.

Suppose the CSP modifies y blocks out of the m outsourced blocks. Then, the probability of corrupted blocks is equal to $p_y = \frac{y}{m}$. Let c be the number of blocks that the auditor asks to verify the integrity of the outsourced data in the challenge step and n be the number of sectors in each block. Let x be a discrete random variable that indicates the number of blocks chosen by the auditor that matches the blocks modified by the CSP. We compute the probability that at least one of the blocks picked by the auditor is matched

with one of the modified block by the server, namely $P_x(x \geqslant 1))$ as follows:

$$\begin{aligned} p_x(x \geqslant 1) &= 1 - p_x(x = 0) \\ &= 1 - \frac{m-y}{m}.\frac{m-y-1}{m-1}\cdots\frac{m-y-c+1}{m-c+1} \\ &= 1 - (1-\frac{y}{m}).(1-\frac{y}{m-1})\cdots(1-\frac{y}{m-c+1}) \\ &= 1 - \prod_{i=1}^{c-1}(1-\frac{y}{m-i}) \end{aligned}$$

Since $(1-\frac{y}{m-i}) \leqslant (1-\frac{y}{m})$, then $1-\prod_{i=1}^{c-1}(1-\frac{y}{m-i} \geqslant 1-(1-\frac{y}{m})^c$. Therefore, the probability of detection $p_x(x \geqslant 1)$ is:

$$p_x(x \geqslant 1) \geqslant (1-\frac{y}{m})^c = 1-(1-p_y)^c \tag{6.2}$$

On the other hand, it is clear that $(1-\frac{y}{m-i}) \geqslant (1-\frac{y}{m-c+1})$, then $1-\prod_{i=1}^{c-1}(1-\frac{y}{m-i}) \leqslant 1-(1-\frac{y}{m-c+1})^c$. Therefore:

$$p_x(x \geq 1) \leqslant 1-(1-\frac{y}{m-c+1})^c \tag{6.3}$$

As a result, from Equation 6.2 and 6.3, we conclude that the probability of detection between $(1-\frac{y}{m})^c$ and $1-(1-\frac{y}{m-c+1})^c$.

From the point of view of sectors, since, each of the block consists of n sectors, such probability on the basis of sector corruption p_s is computed by:

$$\begin{aligned} p_y \geqslant (1-p_s)^n &\Rightarrow 1-(1-p_y)^c \geqslant 1-(1-p_s)^{nc} \\ &\Rightarrow p_x(x \geqslant 1) \geqslant 1-(1-p_s)^{nc} \end{aligned}$$

Suppose the Data Owner (DO) divides 1 GB file into 125000 blocks with size 8

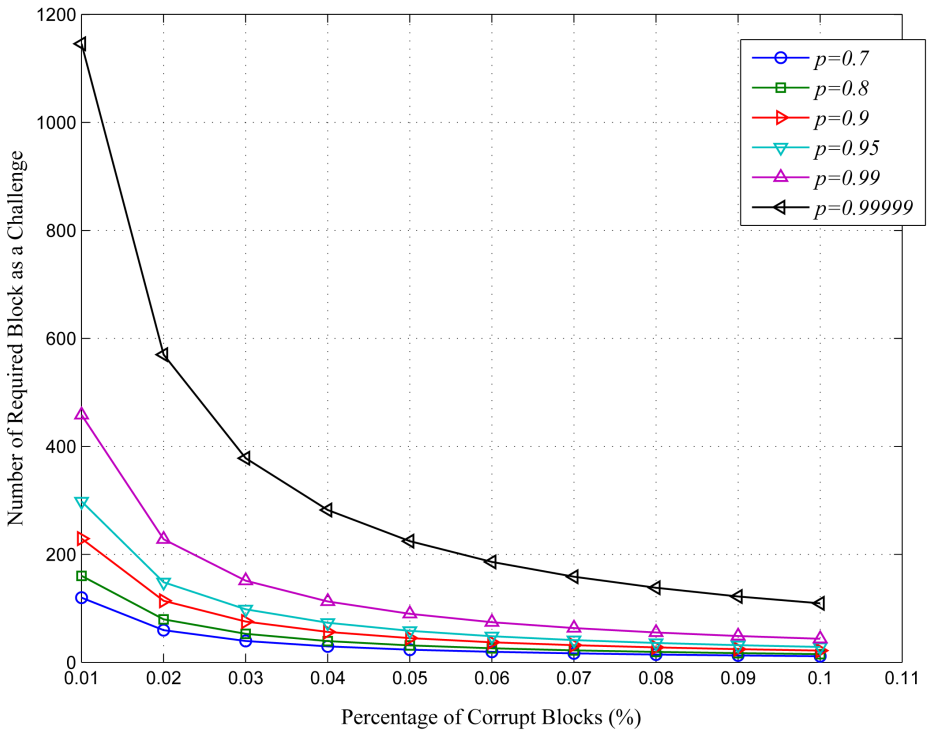


Figure 6.1: Number or required blocks as a challenge message under different number of data corruptions.

KB and outsources the blocks in the cloud after generating the tags. Figure 6.1 shows the number of challenge blocks (c) that are required to detect the different number of corrupted blocks (y) when the probability of misbehavior detection is collected from a set of $P_x = \{0.7, 0.8, 0.9, 0.99, 0.99999\}$. For example, if the server modifies $p_y = 0.01$ of the outsourced blocks $(m = 125000)$, the auditor needs to randomly select 230 blocks as a challenge to achieve P_x of at least 0.90. As it is clear, by increasing the number of corrupted blocks, the least number of challenge blocks are required to achieve such a probability of detection (i.e., only 22 blocks for $p_y = 0.1, P_x = 0.9$).

Figure 6.2 illustrates the number of challenge blocks when the probability of misbehavior detection is between 0.5 and 1 with variable rate of data corruption. For example, if the server modifies 0.01% out of the m outsourced blocks, the DO needs to randomly select 527 data blocks as a challenge for detecting the corrupted blocks with probability of 0.9949. It also can be seen when the rate of corrupted blocks is around 0.1%, the min-

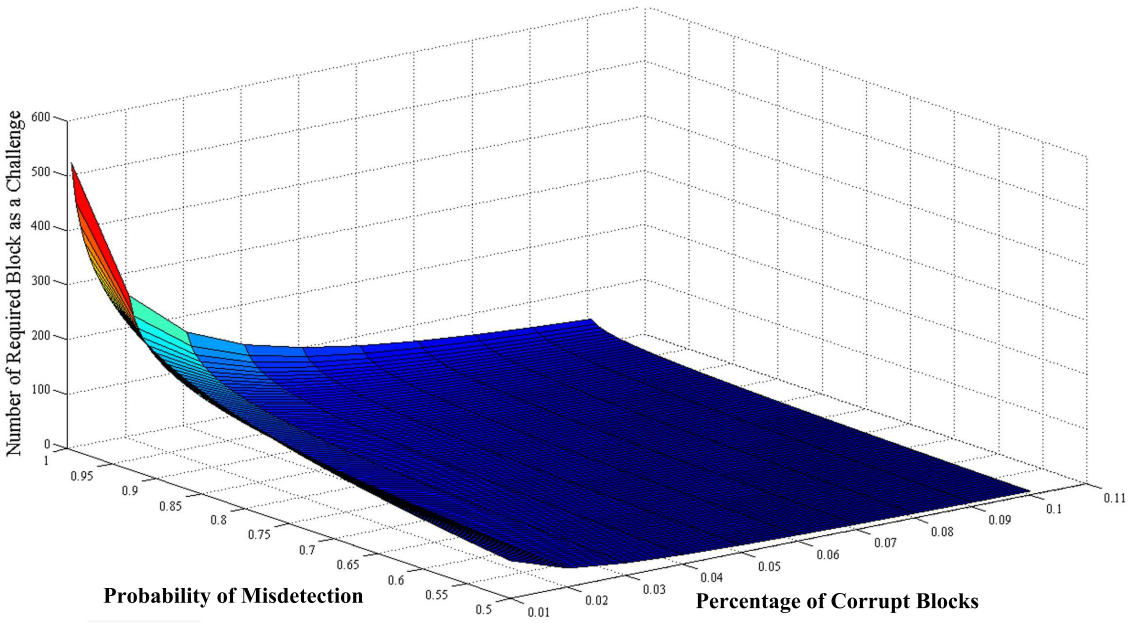


Figure 6.2: Number or required blocks as a challenge message under probability of misbehavior detection is from 0.5 to 1.

imum numbers of challenge blocks (between 34 and 51 data blocks) are required to audit the outsourced data with probability 0.5 to 0.9949.

6.4 Performance Analysis of Processing Time

In this section, we assess the performance of the proposed remote data checking method in terms of the computation burden (processing time) in the setup, challenge, response, and verification phases.

6.4.1 Results of Setup Phase

As previously mentioned in Section 4.2.1.1 of Chapter 4, in the setup phase, the data owner divides the input file into *m* blocks, generates the public and private keys, and computes a unique tag for each block.

Figure 6.3 shows the processing time of setup phase for different size of files when the signature length is 16 in 20 different experiments. When the size of input file is 10 MB, the incurred processing time on the data owner is 757 second. As it is clear, upon increasing the size of file, more processing time is also imposed on the data owner because

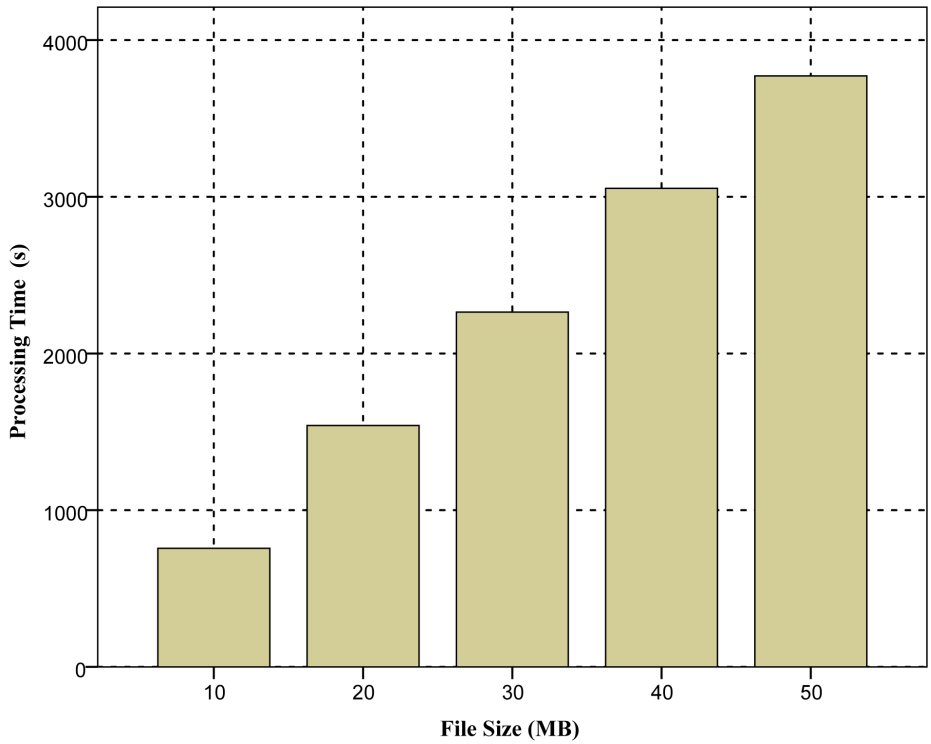


Figure 6.3: The Processing Time of Setup Phase When the Signature Length is 16.

of increasing the number of blocks. For example, the processing time of 50 MB input file is 3771 second. This experiment is repeated when the length of signature is 32 (Figure 6.4), 64 (Figure 6.5), 128 (Figure 6.6), and 256 (Figure 6.7).

The processing time of setup phase for various sizes of the input file is compared on the basis of the signature length (SL). As it is shown in Figure 6.8, by increasing the length of signature from 16 bits to 256 bits, the less computation overhead is incurred on the data owner. It is because, the number of sectors in each block are also increasing. For example, when the size of file is 50 MB, the processing time is in the range from 3771 to 599 second for 16 bit signature and 256 bit signature respectively.

The Setup phase is a one-off time-consuming process, especially when the size of files are big. In other words, most of the time the owner needs to carry out the setup phase for only one time. In Figure 6.9, we show the computation overhead of large-scale files from 1 GB to 10 GB when the length of signature is 256 bits. For instance, the processing time of setup phase for computing tags of 5 GB file is around 61329 seconds.

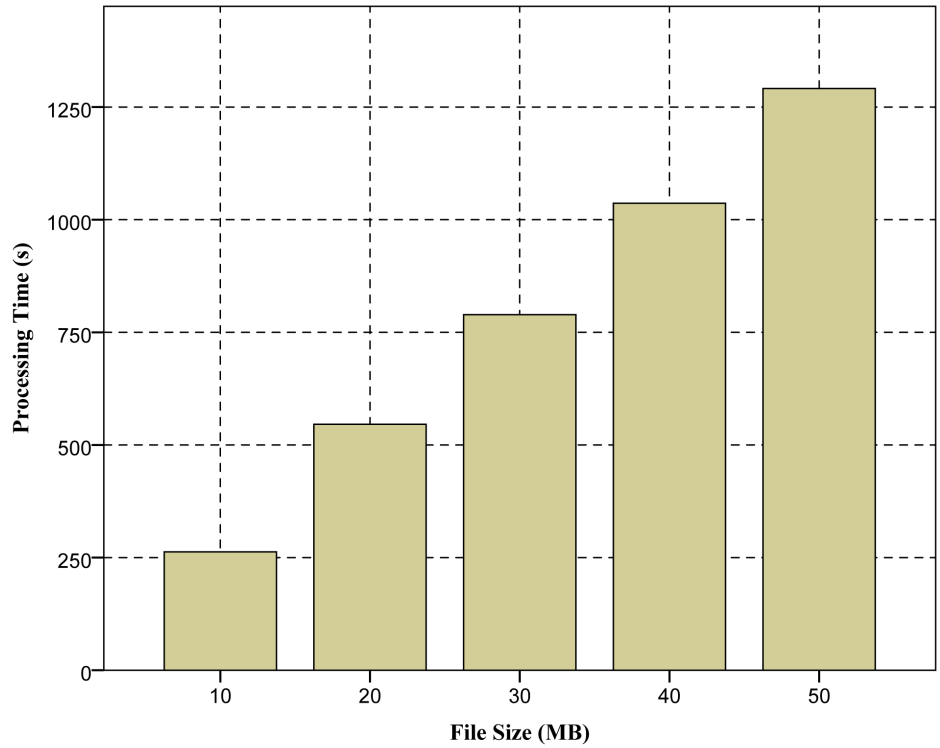


Figure 6.4: The Processing Time of Setup Phase When the Signature Length is 32.

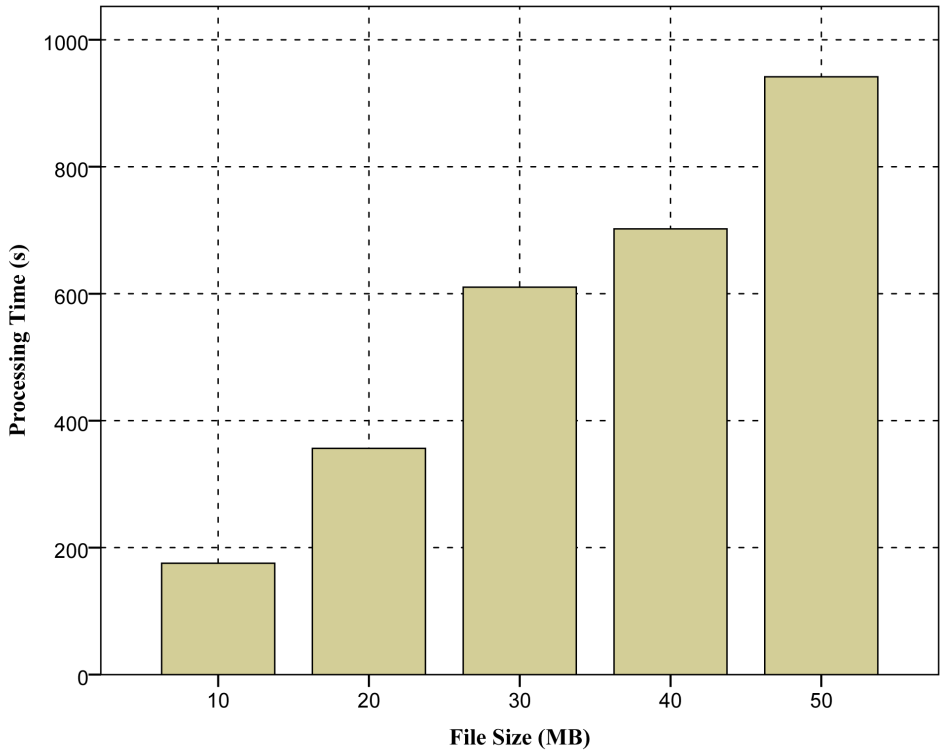


Figure 6.5: The Processing Time of Setup Phase When the Signature Length is 64.

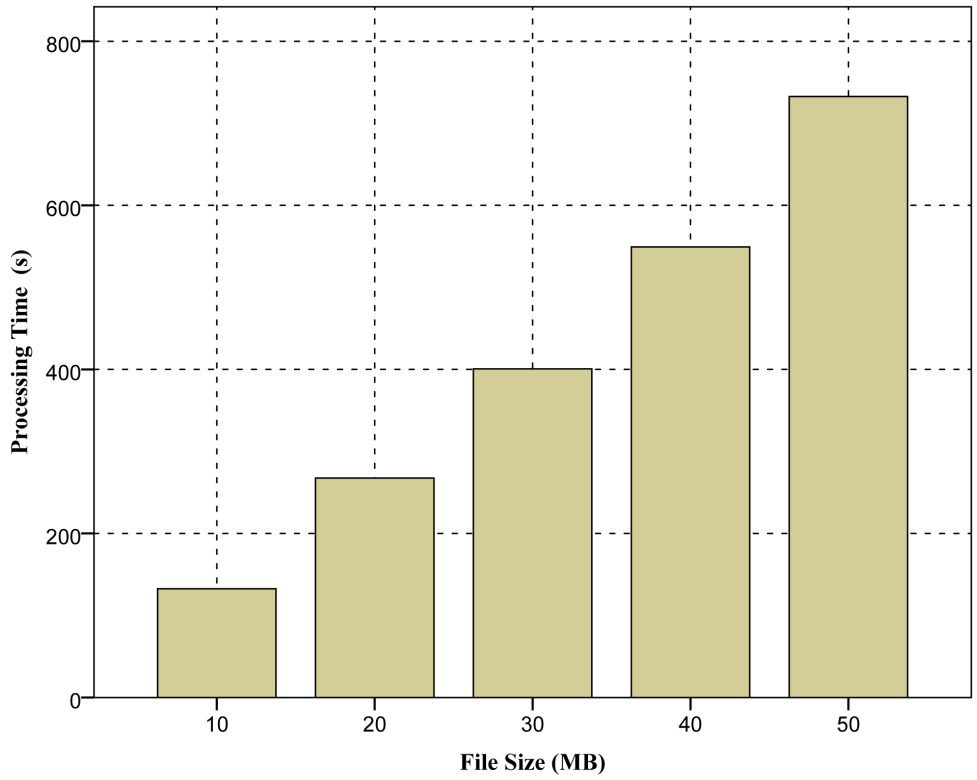


Figure 6.6: The Processing Time of Setup Phase When the Signature Length is 128.

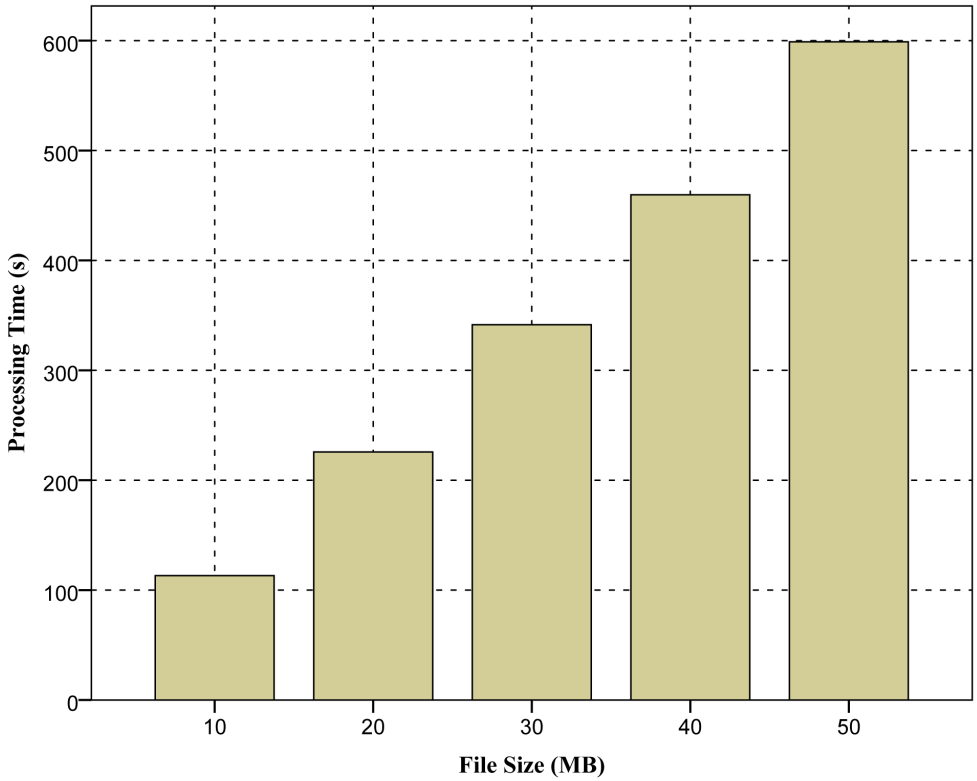


Figure 6.7: The Processing Time of Setup Phase When the Signature Length is 256.

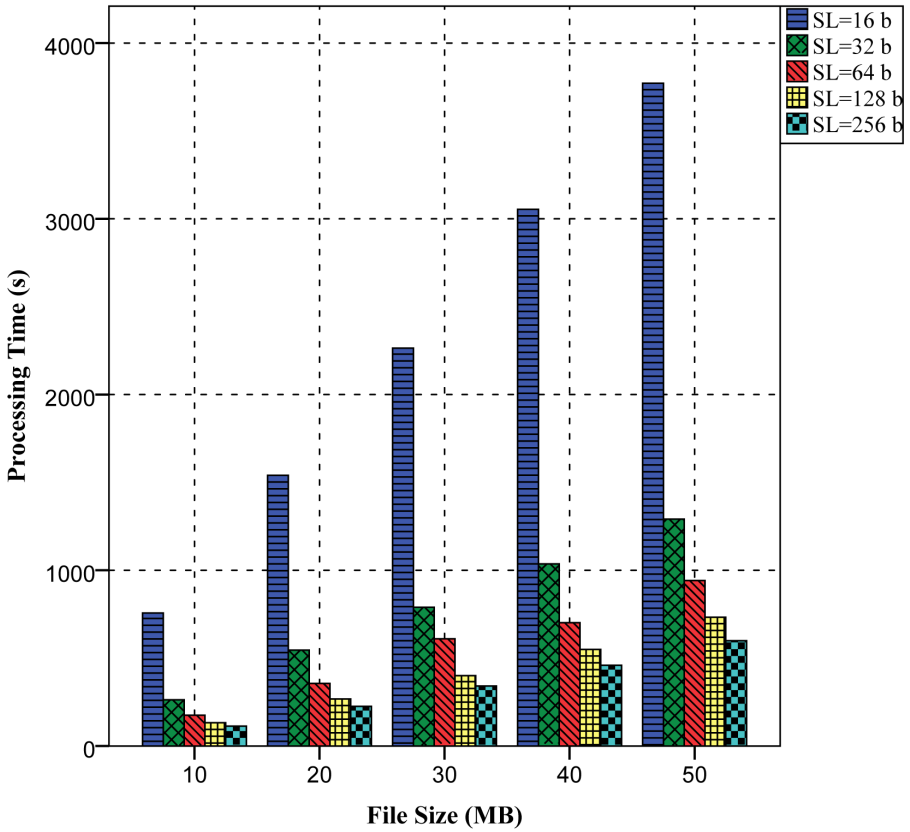


Figure 6.8: The Comparison of Processing Time in Setup Phase Based on the Various Size of Signature.

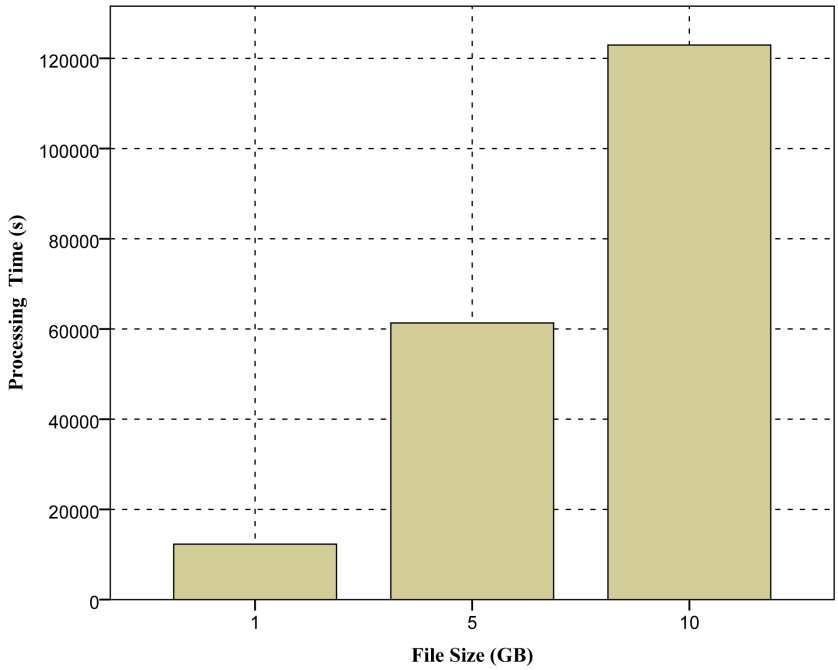


Figure 6.9: The Processing Time of Setup Phase for Large Scale Files When the Signature Length is 256.

6.4.2 Results of Challenge Phase

As mentioned earlier in Section 4.2.1.2 of Chapter 4, the auditor needs to select a number of blocks as a challenge message. In this section, it shows how many computation burdens are incurred on the auditor to achieve the 90%, 95%, and 99% probability of detection.

Figure 6.10 illustrates the processing time for generating the random challenge that consists of c random indices and coefficients $(\{cs_i, v_i\}_{i=1}^{c})$ in 20 different experiments. As it is clear, to achieve 90% probability of detection, the auditor must select 230 blocks out of 2560 in 10 MB, 5120 in 20 MB file, 10240 in 30 MB, 5129 in 40 MB, and 12800 in 50 MB. By increasing the probability of detection, the auditor requires selecting more data blocks as the challenge, which causes more processing time on the auditor. From the other perspective, the processing time of challenge phase is independent from the signature length and directly depends on the probability of detection. The graph also shows that the size of file has a negligible effects on the processing time of this phase.

The processing time of generating the challenge messages for large-scale files is also displayed in Figure 6.11 in which the number of blocks is 262144, 1310720, and 2621440. The graph shows that the processing time of the challenge messages for 90% probability is between 12.8 and 13.6 ms. When the probability of detection reaches 95%, the processing time is in the range of 15.73 to 16.4 ms. Finally, by increasing the probability of detection to 99%; the processing time of challenge phase is also increasing and fluctuating between 17.2 to 18.73 ms. Therefore, the growth rate of processing time is independent from the size of file during the challenge phase, and the auditor is able to generate the challenge massage several times with such cost. This is because the auditor only needs to choose c random blocks, which incurs a small cost on the auditor.

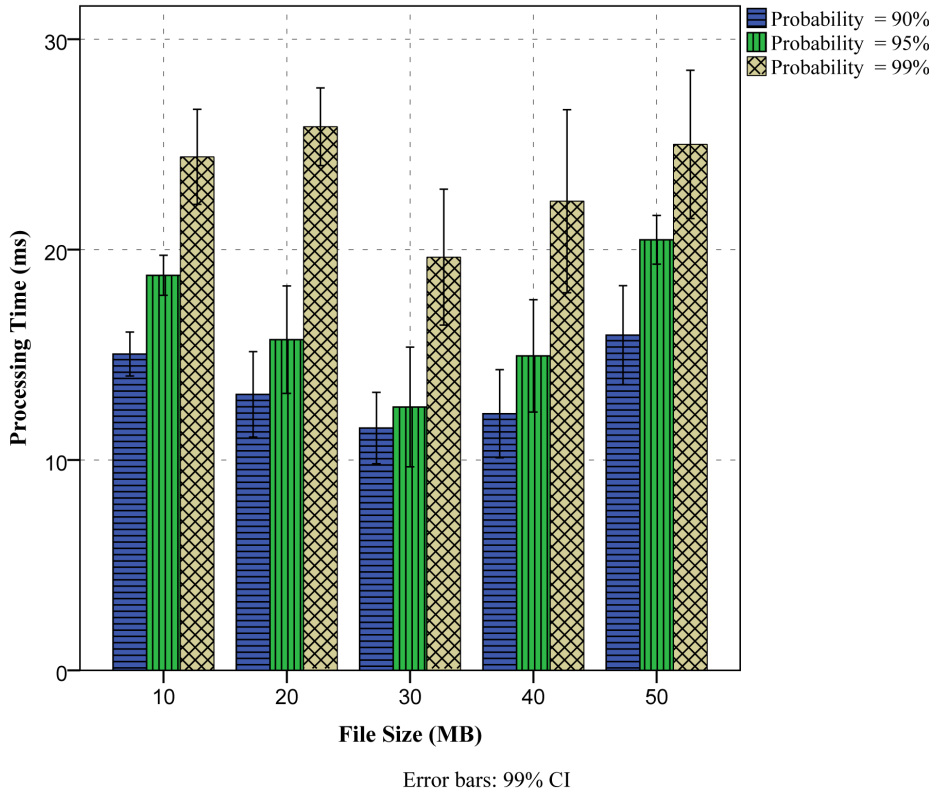


Figure 6.10: The Processing Time of Challenge Phase for different size of the file.

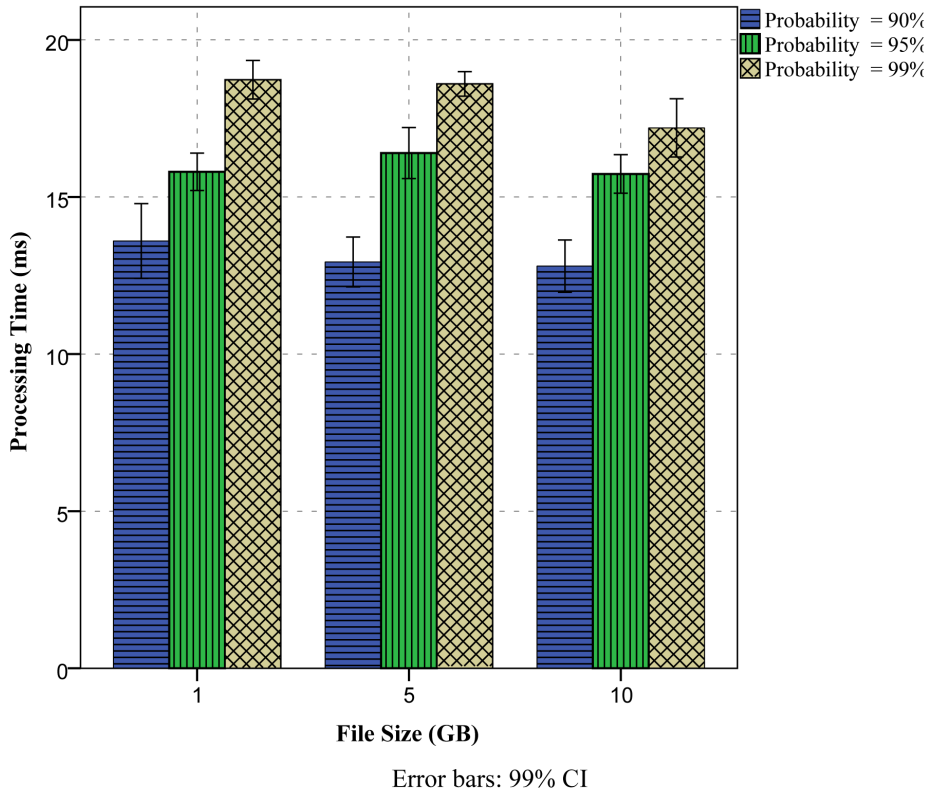


Figure 6.11: The Processing Time of Challenge Phase for Large-Scale Files.

6.4.3 Results of Response Phase

When the server receives the challenge message, the response message is computed on the basis of the received challenge, the stored blocks, and tags (the details can be found in Section 4.2.1.2 of Chapter 4). Therefore, the processing time of the response phase is incurred on the server. This section presents implementation results of response phase with probability of detection 90%, 95%, and 99%.

Figure 6.12 shows the impact of file size on the processing time of the response messages for 16 bit signature. When the auditor sends a challenge message, including 230 blocks to achieve 90% probability, the computation time changes from 7.56 to 7.99 second. For 95% probability of detection, the computation time is around 10 seconds and for 99% probability, the computation time is increasing from 15.24 to 15.87 second. Therefore, it can be concluded that the computation time is roughly constant and is independent from size of file.

The impact of probability of detection on the computation time of the response messages is also evaluated when the length of signature is 16 bits. Figure 6.13 demonstrates that by increasing the probability of detection, the number of the requested blocks are also increasing, which caused additional processing time to be incurred on the prover. For example, the imposed computation on the server during the response phase of 10 MB file, are 7.56 second for 90% probability. When the probability of detection increases to 95% the processing time is also increasing to 9.95 second. In 99% probability, the processing time reaches the 15.24 second. The graph also shows that the processing time of 50 MB file reaches 7.99 s, 10.22 s, and 15.73 s for 90%, 95%, and 99% probability of detection respectively.

Figure 6.14 illustrates the impact of normal file size on the computation time during the response phase with 32 bits as the length of signature. As it is clear, when the

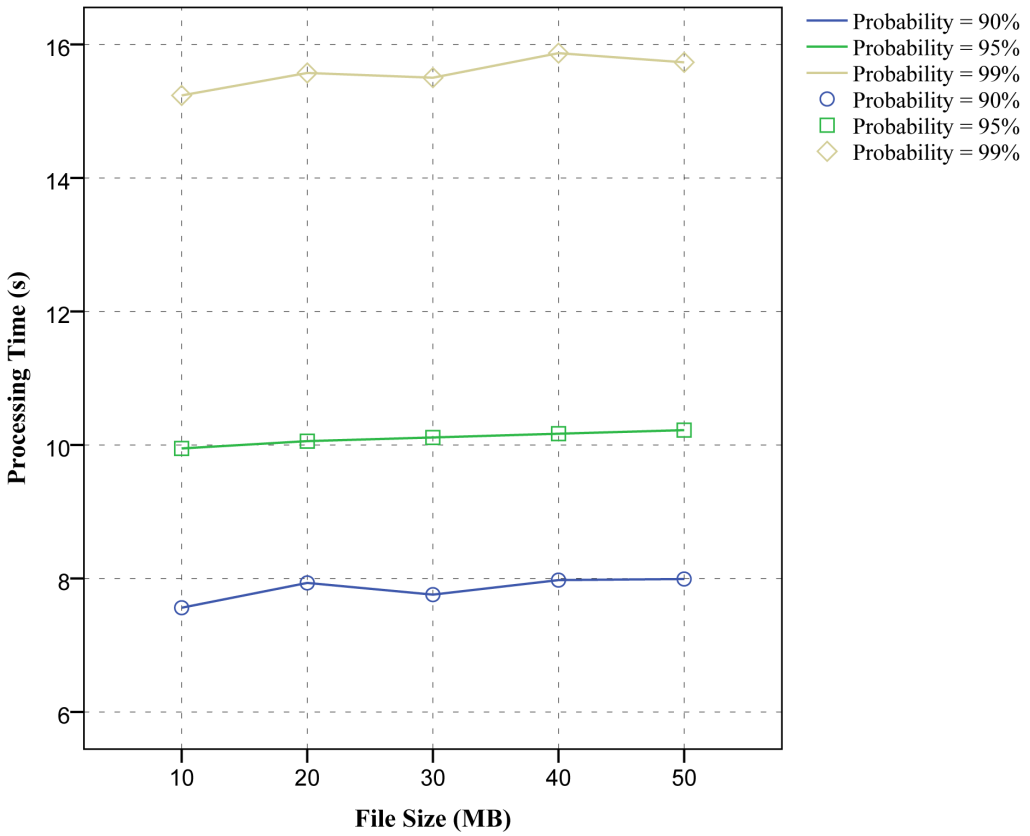


Figure 6.12: The Impact of Normal File Size on Processing Time during Response Phase when signature length is 16.

probability of detection is 90%, the processing time fluctuates between 7.65, and 8.22 s. By augmenting the probability to 95%, the processing time varies slightly from 9.91 s to 10.49 s. Finally, a slight increase cost is also observed during the response phase of normal file size with 99% probability (between 15.19 s, and 15. 78 s).

The effect of the different probability of detection on the processing time of response phase for normal file size with 32 bit signature is shown in Figure 6.15. It can be seen when the size of file is 10 MB, the processing time is about 7.65 s, 9.91 s, and 15.19 s for 90%, 95%, and 99% probability of detection respectively. By augmenting the size of file, the computation time of response phase is slightly increasing, for example; the processing time in 50 MB file size is 8.22 s, 10.49 s, and 15.78 s on the basis of the amount of detection probabilities.

Figure 6.16 displays the impact of the file size on the processing time of response

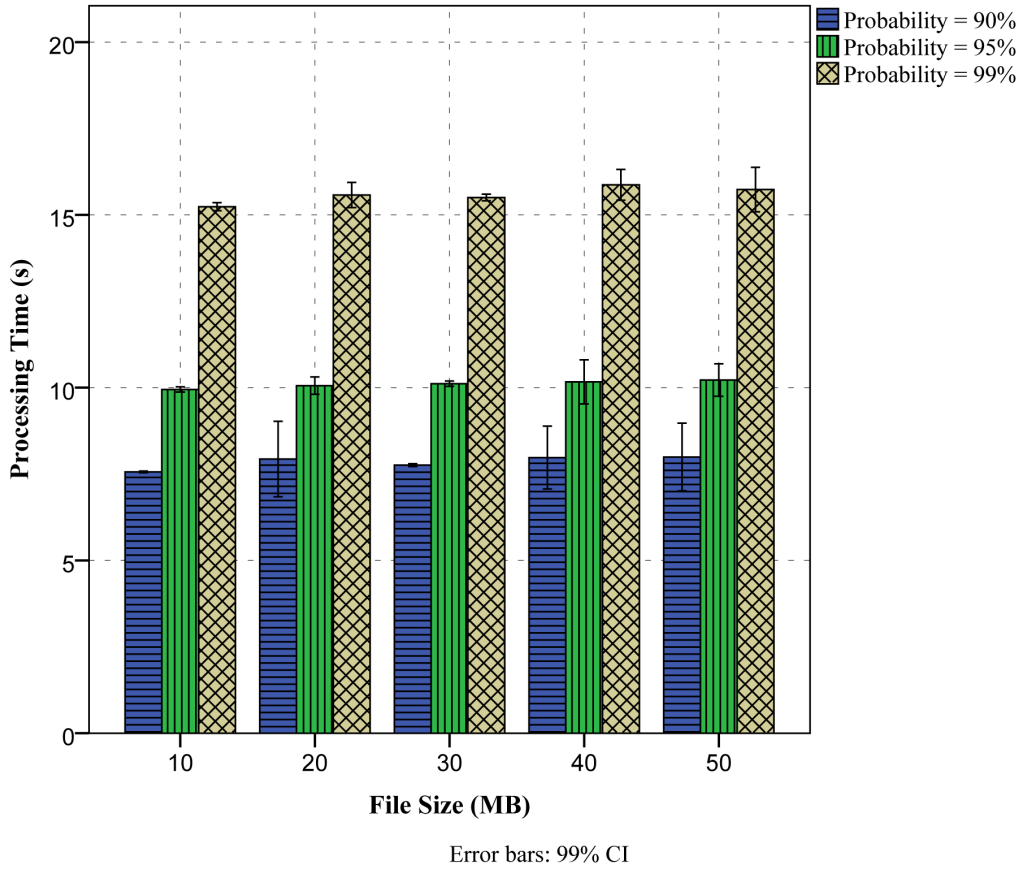


Figure 6.13: The Impact of Probability of Detection on Processing Time during Response Phase for Normal File Size When Signature Length is 16.

phase for 64 bit signature. It can be seen that the size of file has an inconsiderable effect on the processing time of the response phase. For instance, when the probability of detection is 90%, 95%, or 99%, the processing time is about 7.85 s, 1026 s, or 15.48 s respectively.

The effect of probability of detection on processing time is also illustrated in Figure 6.17 for 64 bit probability. The computation time for the file with size 10 MB, is around 7.68 s, 9.89 s, and 15.19 s for different probability of detection. When the size of file reaches 50 MB, the processing time approaches to 7.84 s, 10.49 s, and 15.81 s regarding to probability of detection.

The effect of file size on the processing time for 128 bit signature is analyzed in Figure 6.18. This graph demonstrates that the size of file has not been considerable influence on the processing time of response phase. For example, the processing time for 90%, 95%, and 99% of the probability is approximately 7.75 s, 10.52 s, and 15.84 s orderly.

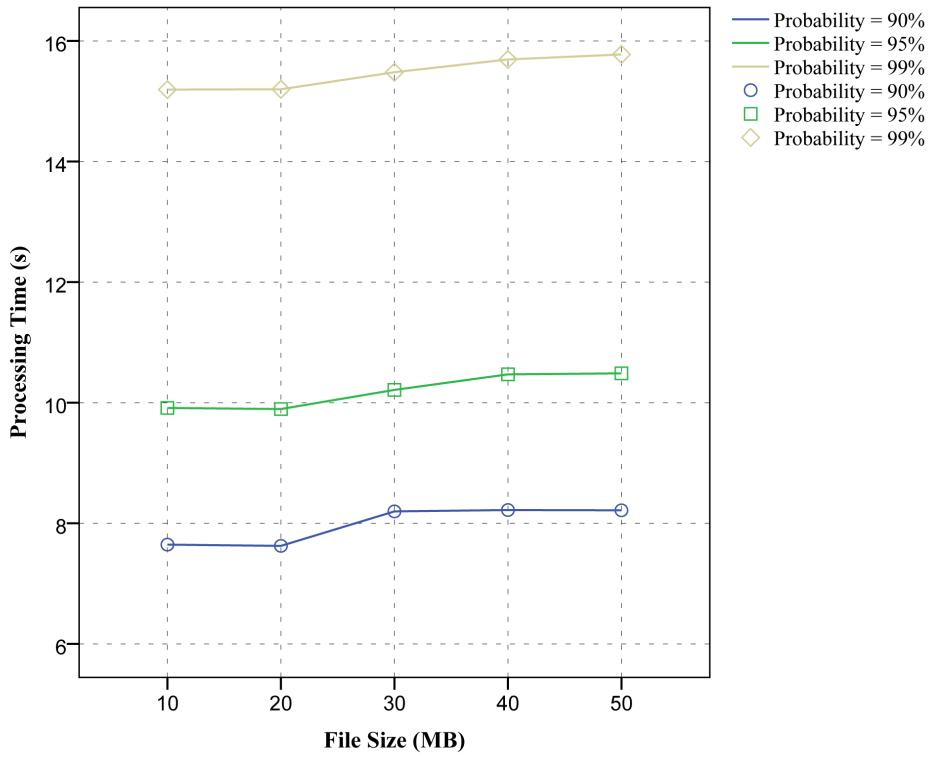


Figure 6.14: The Impact of Normal File Size on Processing Time during Response Phase when signature length is 32.

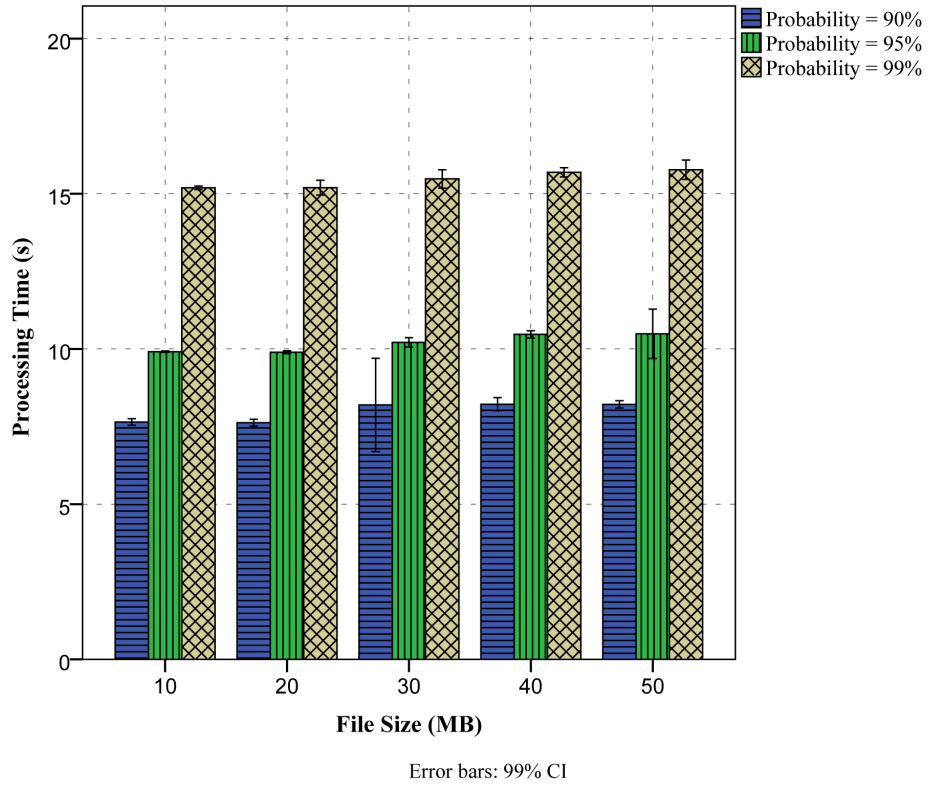


Figure 6.15: The Impact of Probability of Detection on Processing Time during Response Phase for Normal File Size When Signature Length is 32.

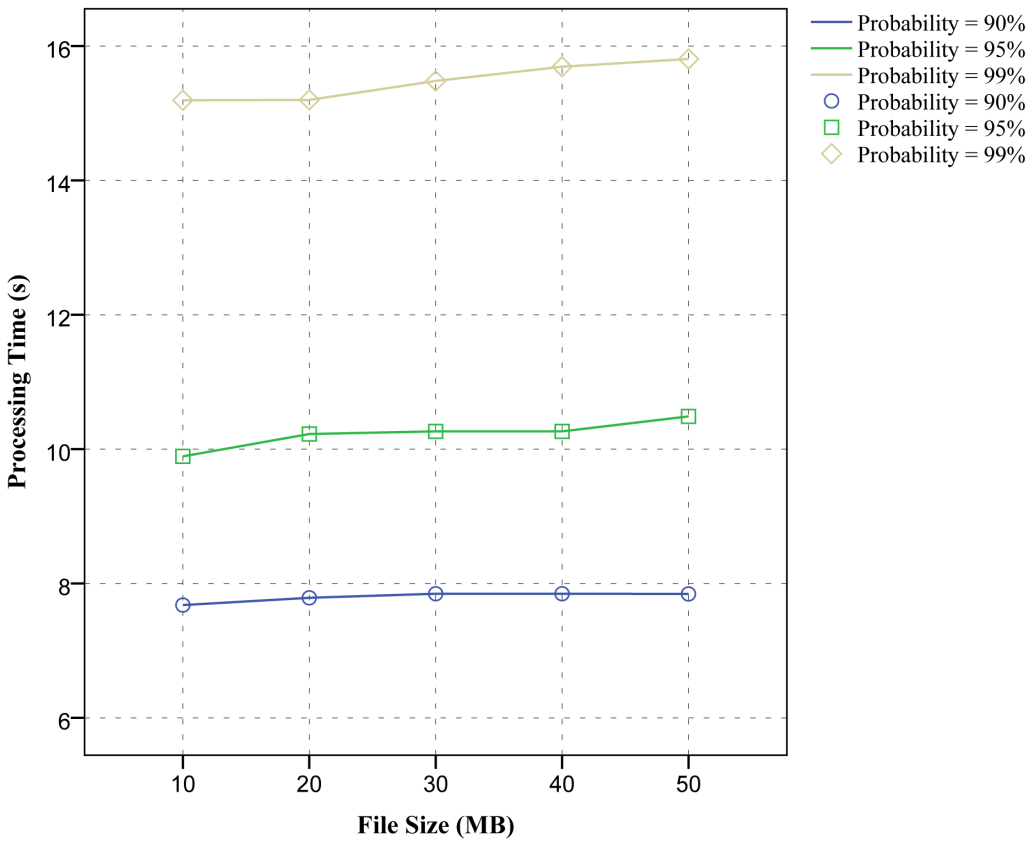


Figure 6.16: The Impact of Normal File Size on Processing Time during Response Phase when signature length is 64.

The impact of probability of detection on processing time of response phase for 128 signatures is revealed in Figure 6.19. Regarding to this graph, when the size of file is 10 MB, the processing time is around 7.68 s, 9.92 s, and 15.26 s for different rate of probability. As such, the processing time for 50 MB file size is 7.94 s, 10.60 s, and 16.12 s respectively.

The processing time of the response phase is also analyzed for 256 bit signature to show the impact of file size on it in Figure 6.20. Similar to Figure 6.18, the processing time for different file size is approximately constant. For example, when the probability of detection is 90%, 95%, or 99%, the processing time of normal file size is around 7.58 s, 10.26 s, and 15.57 s orderly.

The effect of probability of detection on the processing time of response phase is also evaluated in Figure 6.21 for 256 bit signature. Clearly, the processing time depends

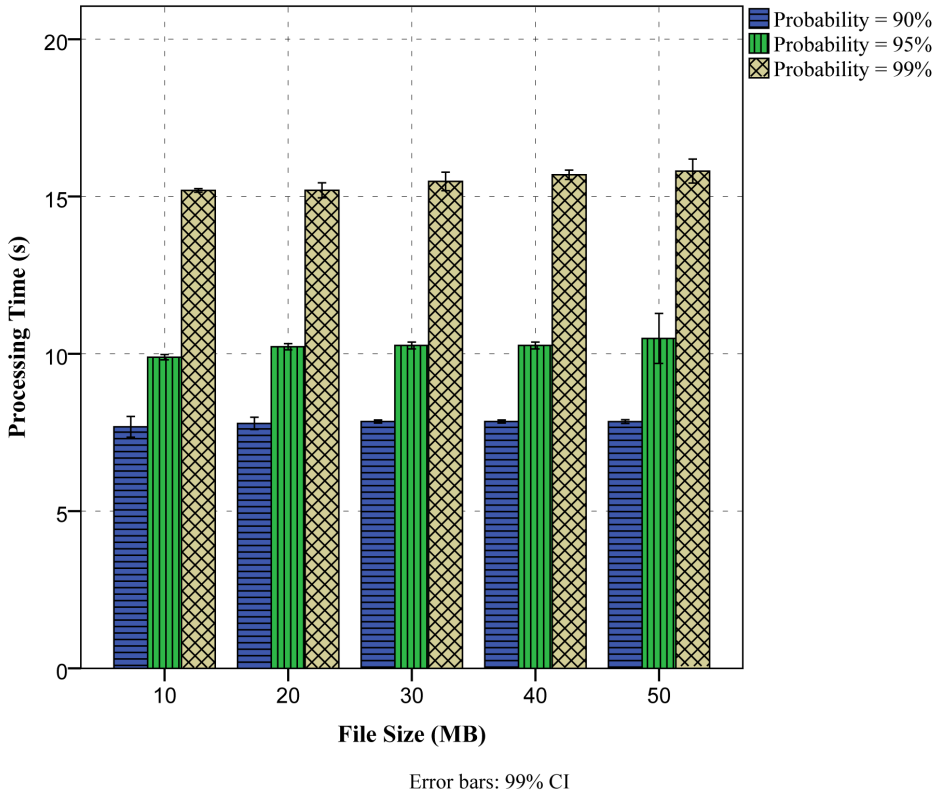


Figure 6.17: The Impact of Probability of Detection on Processing Time during Response Phase for Normal File Size When Signature Length is 64.

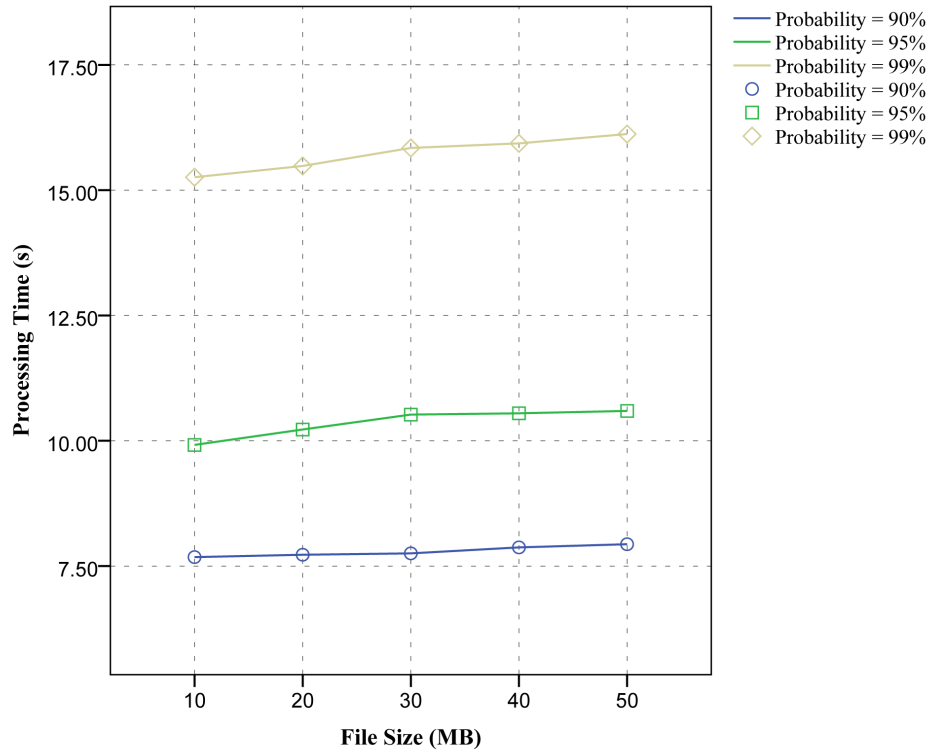


Figure 6.18: The Impact of Normal File Size on Processing Time during Response Phase when signature length is 128.

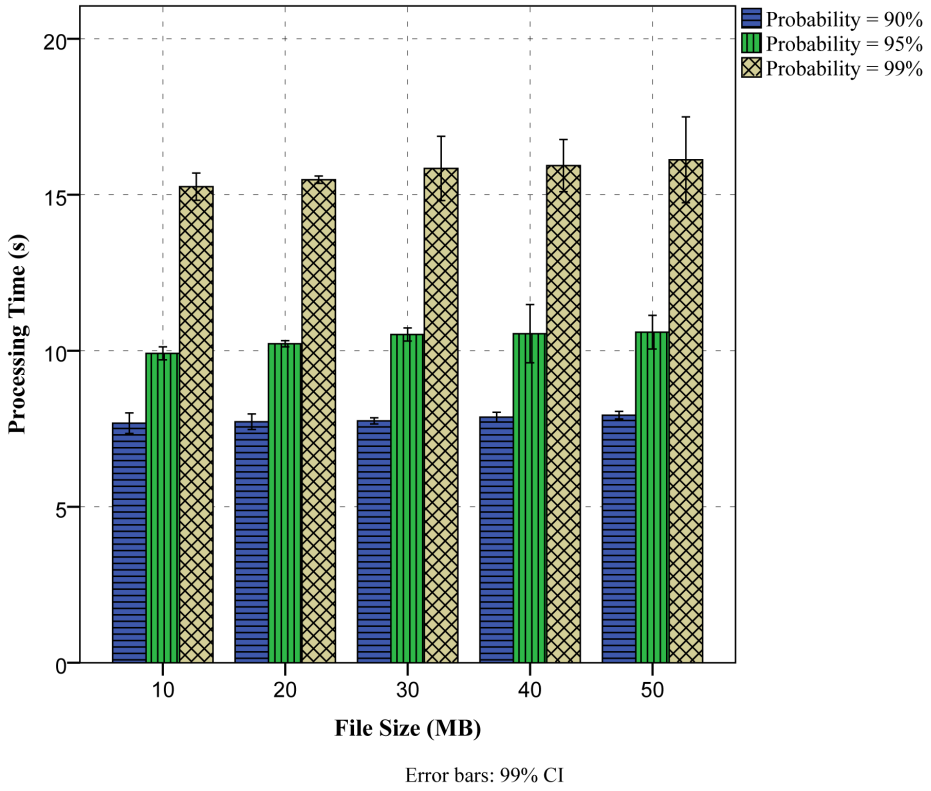


Figure 6.19: The Impact of Probability of Detection on Computation Time during Response Phase for Normal File Size When Signature Length is 128.

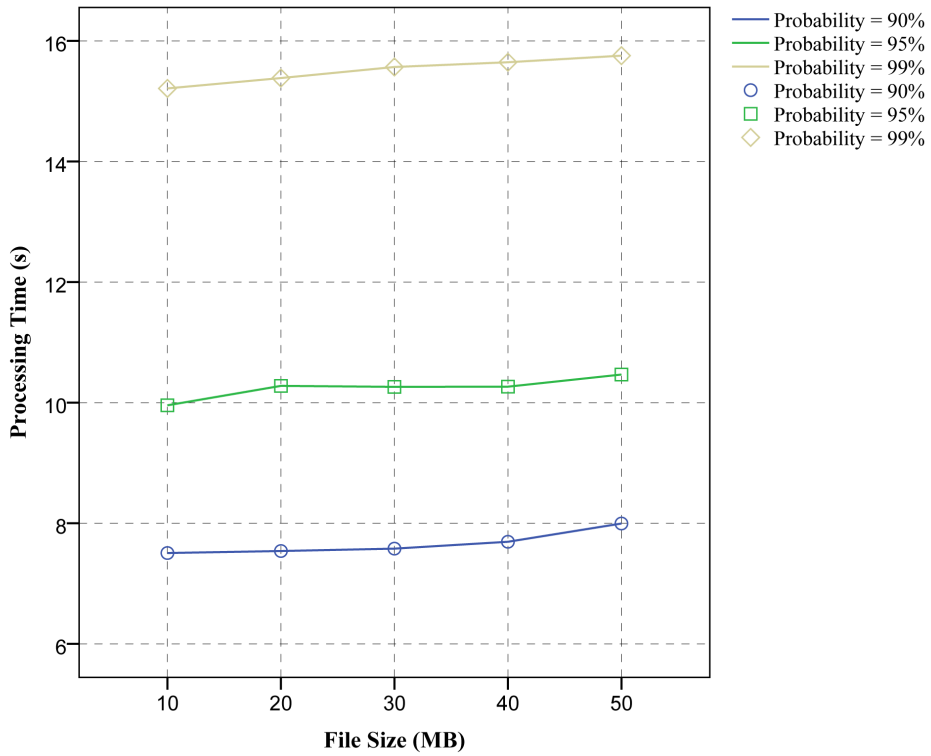


Figure 6.20: The Impact of Normal File Size on Computation Time during Response Phase When Signature Length is 256.

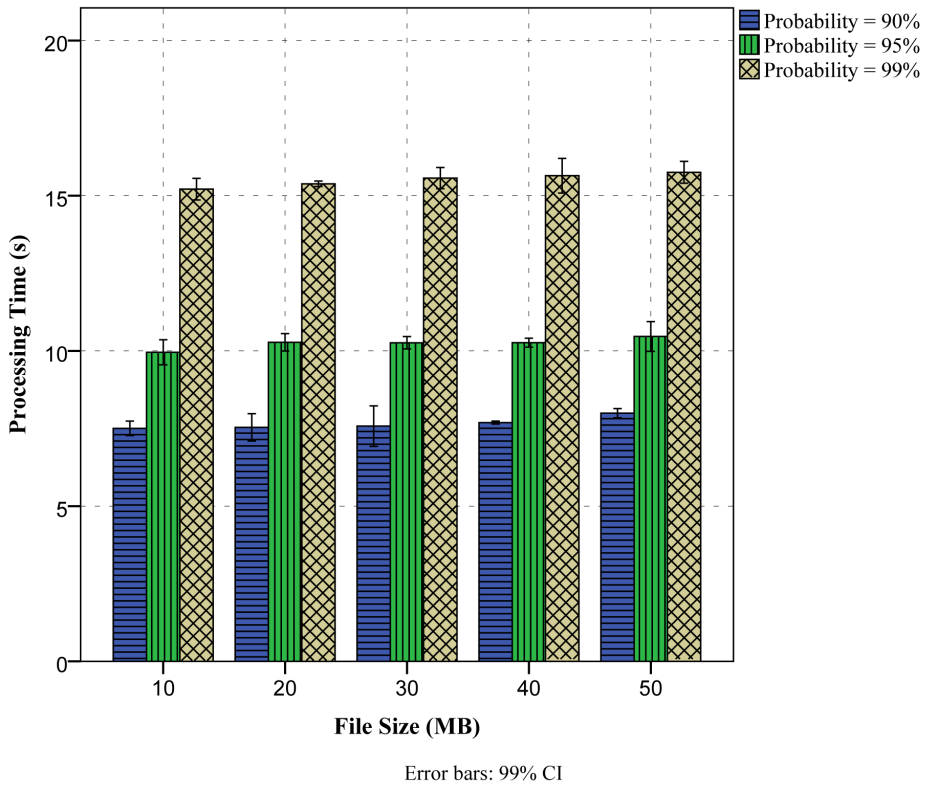


Figure 6.21: The Impact of Probability of Detection on Processing Time during Response Phase for Normal File Size When Signature Length is 256.

on probability of detection directly.

Figure 6.22 compares the processing time of response phase for different size of signature and probability of detection when the size of input file is from 10 MB to 50 MB. The graph demonstrates that the processing time of the response phase only depends on the probability of detection. Moreover, the length of signature has a negligible effects on the processing time of this step.

The processing time experiment of the response phase is also conducted for the large-scale file size of input file (1, 5, and 10 GB) to prove that the processing time of response phase is independent of the size of the file. Figure 6.23 displays that the computation time of such files are around 7.8, 10.6, and 16.1 when the probabilities of detection are 90%, 95%, and 99% respectively. Moreover, Figure 6.24 shows the impact of probability of detection on processing time during the response phase for large-scale file sizes. As it is clear, by increasing the number of the probability, the processing time is also increasing.

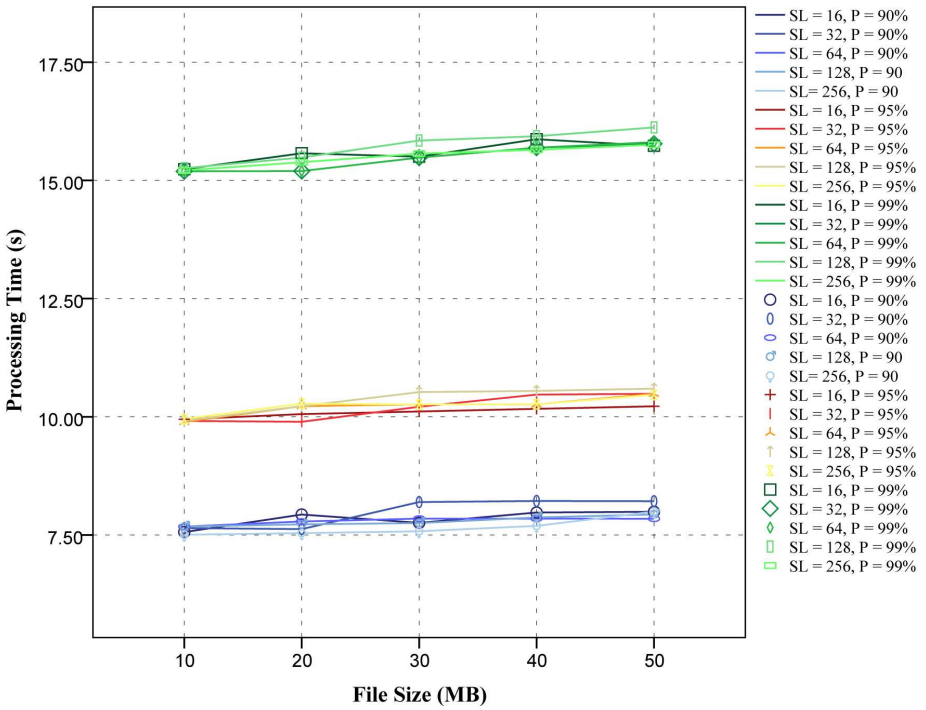


Figure 6.22: The Comparison of Processing Time for Different Normal File Size, Probability of Detection, and Signature Length during the Response Phase.

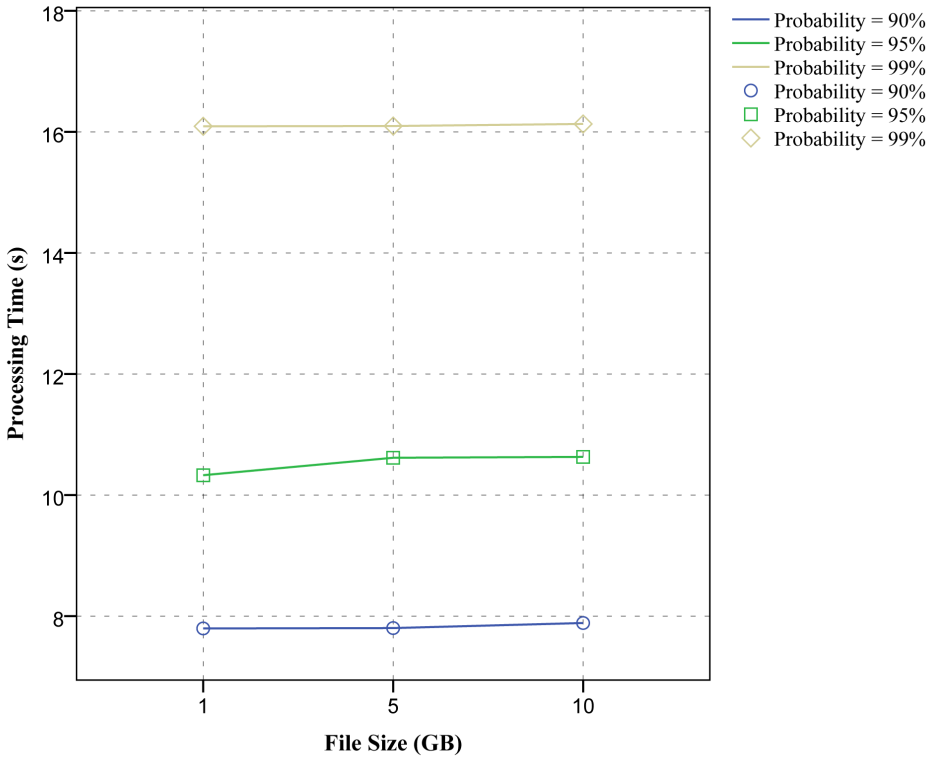


Figure 6.23: The Impact of Large Scale Files on Processing Time during Response Phase When Signature Length is 256.

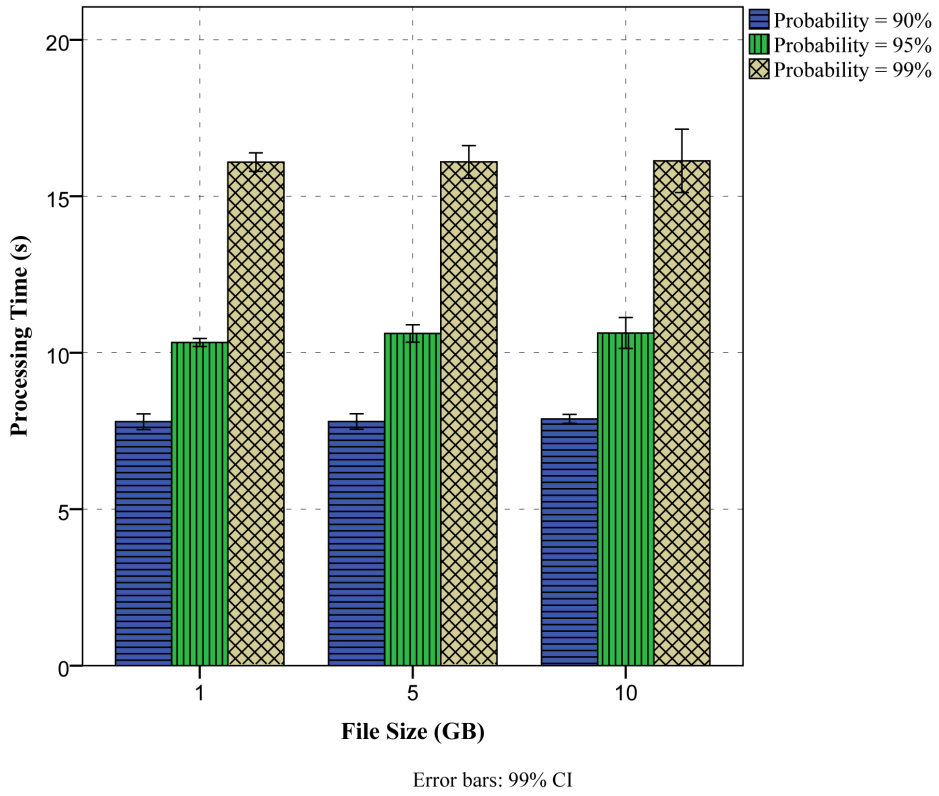


Figure 6.24: The Impact of Probability of Detection on Processing Time during Response Phase for Large Scale Files When Signature Length is 256.

6.4.4 Results of Verification Phase

The last phase of the proposed data auditing scheme is verification in which the auditor checks the integrity of the outsourced blocks on the basis of the challenge and the response message. Since the verification phase is carried out several times, and the processing time of such phase is directly incurred on the auditor, verification is one of the most important phases. The details of verification phase can be found in Section 4.2.1.3 of Chapter 4. In the rest of this section, we show the processing time of verification phase by using various criteria.

Figure 6.25 illustrates the impact of file size on the processing time of verification phase on the auditor for 16 bit signature. When the probability of detection is 90%, the computation time is around 230 ms. By increasing the probability of detection to 95% and 99%, the computation time rises to 233 ms and 245 ms respectively. The graph also shows that increasing the file size has a tangible effect on the processing time. It is because that

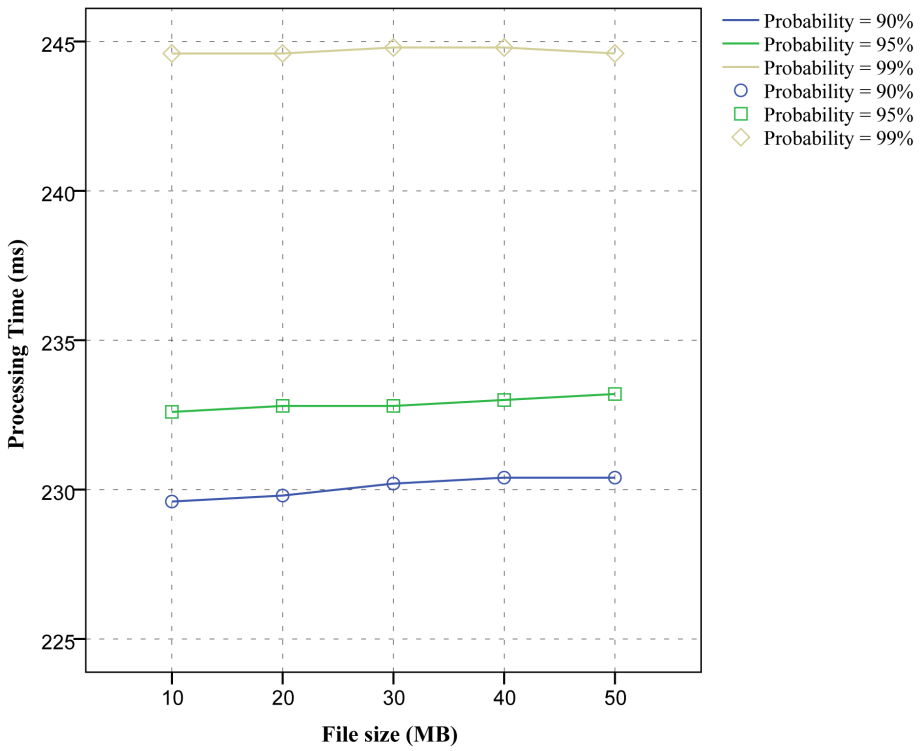


Figure 6.25: The Impact of Normal File Size on Processing Time during Verification Phase When the Signature Length is 16.

the verification of the outsourced blocks is only checked on the basis of the response and challenge messages without requiring the content of the original blocks.

The effect of probability of detection on the processing time of the verification phase is analyzed in Figure 6.26 for 16 bit signature. When the size of input file is 10 MB, the processing time of verification phase for 90%, 95%, and 99% probability is about 229.60 ms, 232.60 ms, and 244.60 ms respectively. The graph also shows that by augmenting the file size to 50 MB, the processing time raised slightly to 230.40 ms, 233.20 ms, and 244.60 ms.

The computation time of verification phase for 32 bit signature is displayed in Figure 6.27 to demonstrate the effect of file size. The result shows that the file size has not been the considerable impact on the variation phase. For example, the computation time for 90%, 95%, and 99% probability is about 64 ms, 69 ms, and 70 ms orderly.

The effect of probability of detection on verification phase with 32 bit signature is

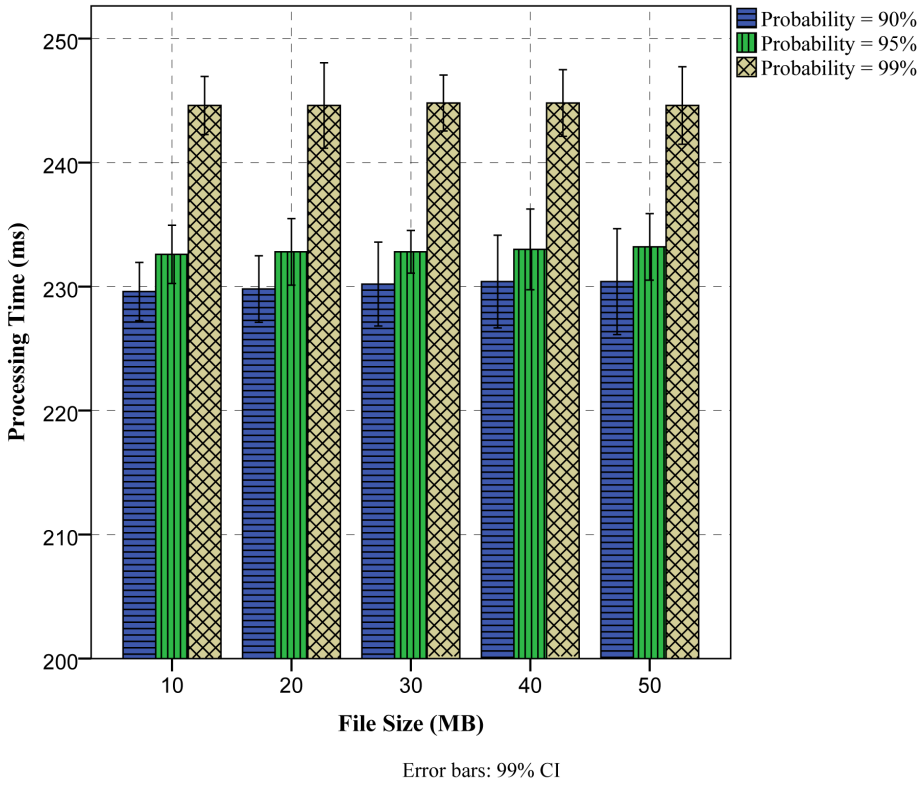


Figure 6.26: The Impact of Probability of Detection on Processing Time during Verification Phase for Normal File Size When Signature Length is 16.

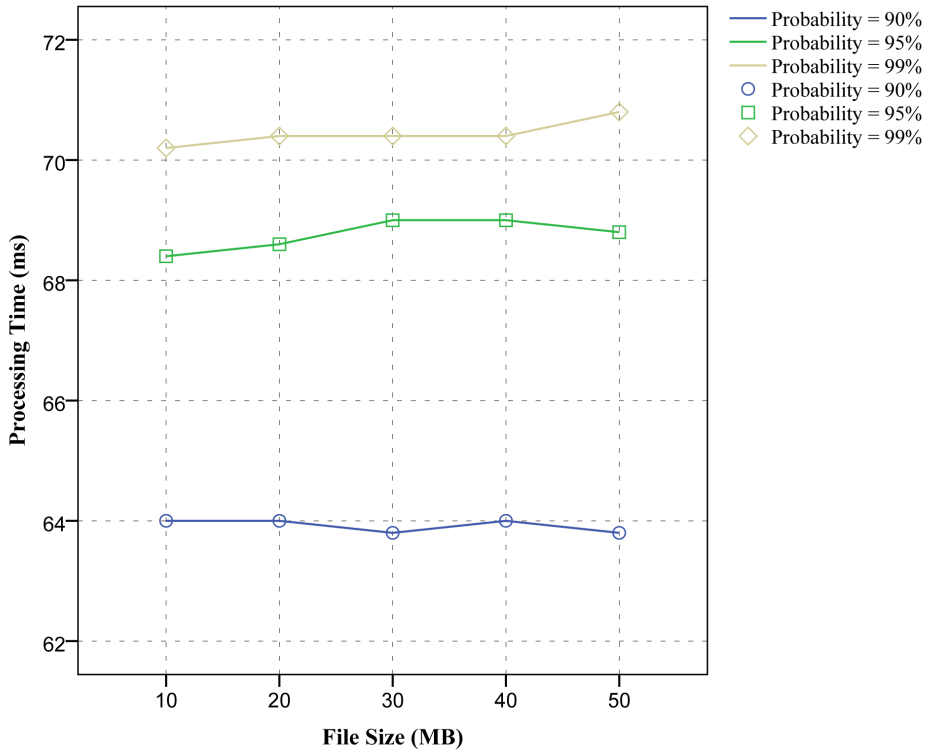


Figure 6.27: The Impact of Normal File Size on Processing Time during Verification Phase When the Signature Length is 32.

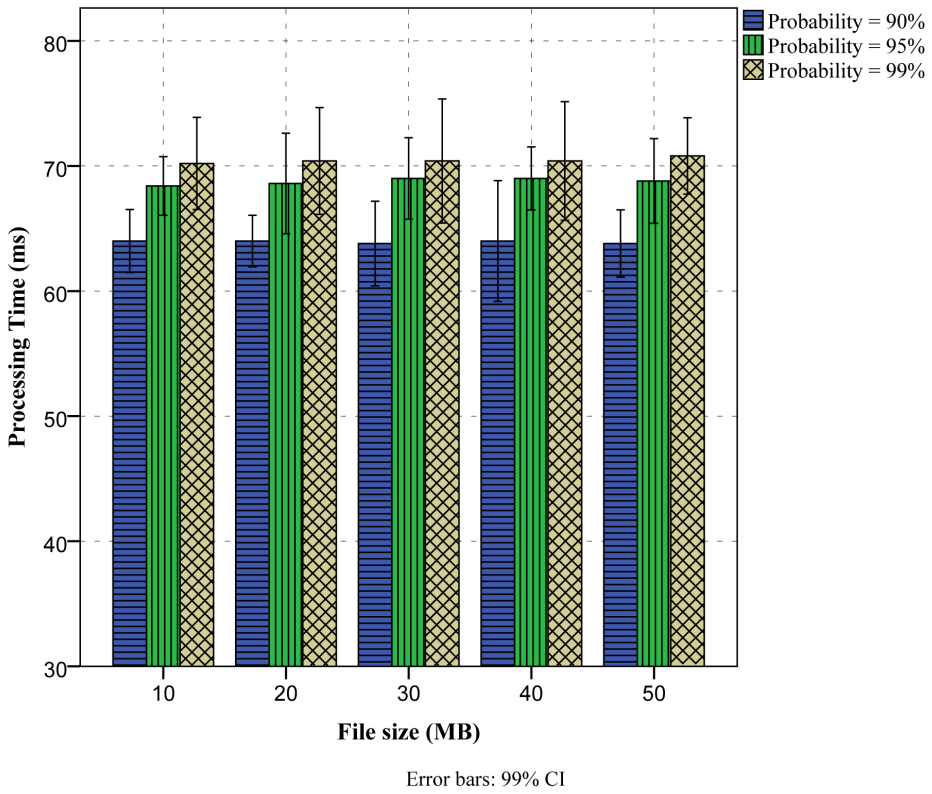


Figure 6.28: The Impact of Probability of Detection on Processing Time during Verification Phase for Normal File Size When Signature Length is 32.

shown in Figure 6.28. When the size of file is 10 MB, the imposed processing time on the auditor for 90%, 95%, and 99% probability is 64 ms, 68.40 ms, and 70.20 ms orderly. Such cost for 50 MB file is around 63.80 ms, 68.80 ms, and 70.80 on the basis of the rate of detection.

Figure 6.29 explains the impact of file size on processing time of verification phase with 64 bit signature. This graph also proves that the rate of such cost is approximately constraint and is independent from the size of file (31 ms for 90% probability, 32 ms for 95% probability, and 35 ms for 99% probability of detection).

Figure 6.30 shows the effect of probability of detection on processing time of verification phase with 64 bit signature. Clearly the processing time of verification is around 30.80 ms, 32 ms, and 34.80 ms for 90% 95%, and 99% probability of 10 MB file. When the size of file raise to 50 MB, the computation time approached 30.80 ms, 32.20, and 35.20 ms for different rate of detection.

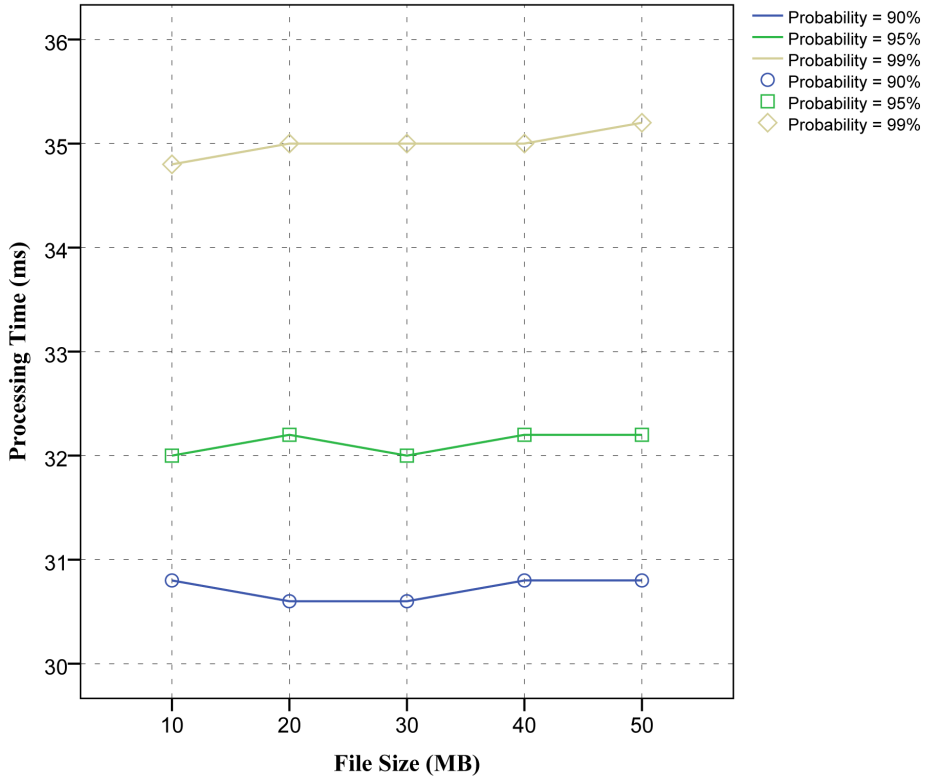


Figure 6.29: The Impact of Normal File Size on Processing Time during Verification Phase When the Signature Length is 64.

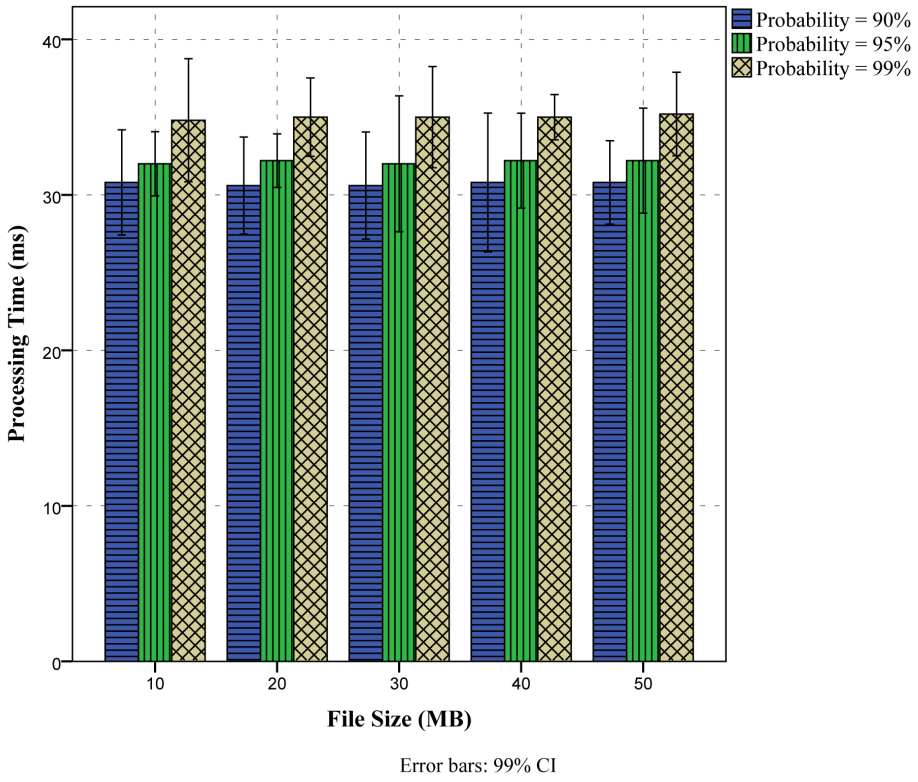


Figure 6.30: The Impact of Probability of Detection on Processing Time during Verification Phase for Normal File Size When Signature Length is 64.

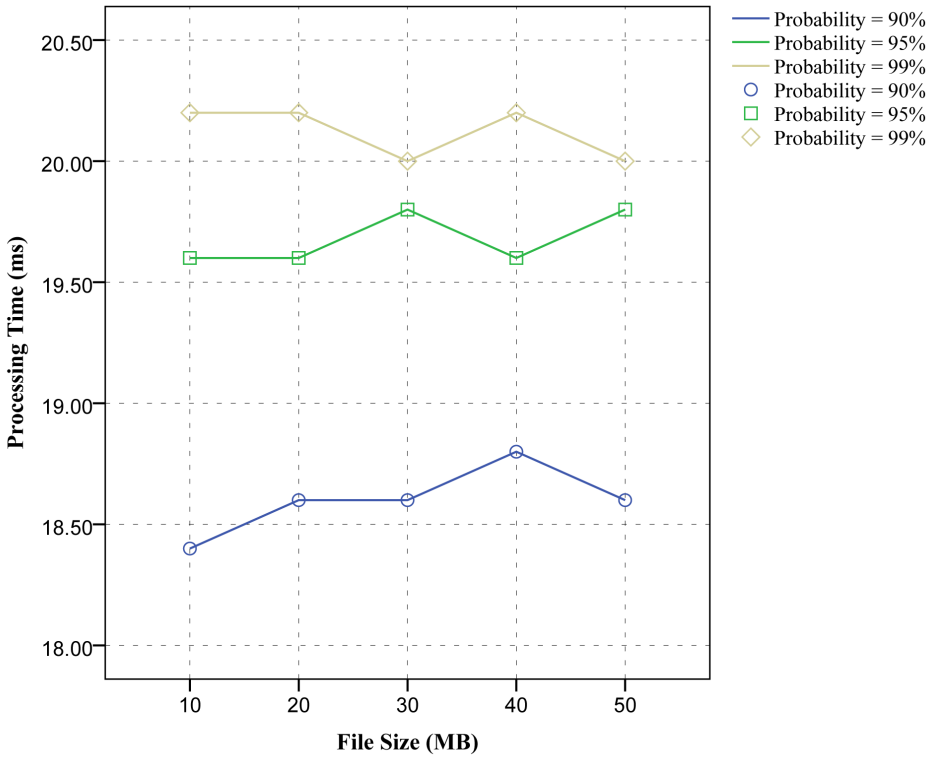


Figure 6.31: The Impact of Normal File Size on Processing Time during Verification Phase When the Signature Length is 128.

The effect of file size and probability of detection on processing time of verification phase with 128 bit signature are illustrated in Figure 6.31 and Figure 6.32 respectively. Figure 6.31 demonstrate that such cost has an insignificant fluctuation (from 18.40 ms to 18.80 ms for 90% probability, 19.60 ms to 19.80 ms for 95% probability, and 20 ms to 20.20 ms for 99% probability).

In Figure 6.32, it is clear that the probability of detection has a direct relation with processing time of verification phase. For example, when the size of file is 10 MB, the computation time is around 18.40 ms, 19.60 ms, and 20.20 ms on the basis of probability of detection.

We conducted such experience to analyze the effect of file size and probability of detection on processing time of verification phase for 256 bit signature in Figure 6.33 and Figure 6.34. From Figure 6.33, it can be understood that the computation time is independent from the file size, and it is about 11.80 ms, 12.60 ms, and 13.40 ms for 90%,

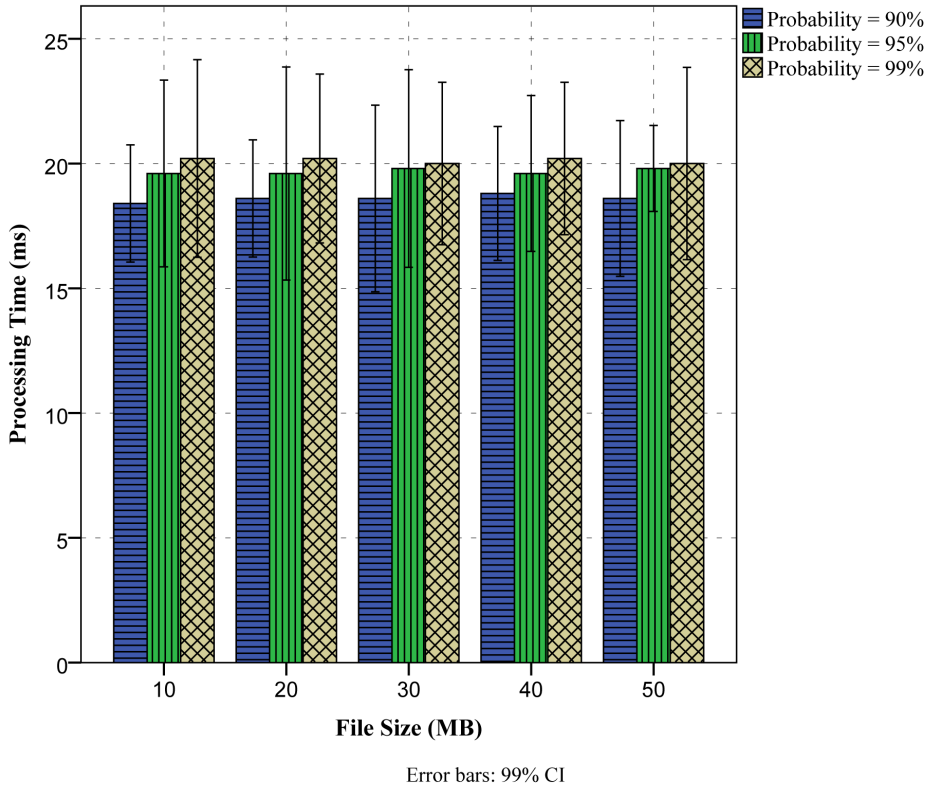


Figure 6.32: The Impact of Probability of Detection on Processing Time during Verification Phase for Normal File Size When Signature Length is 128.

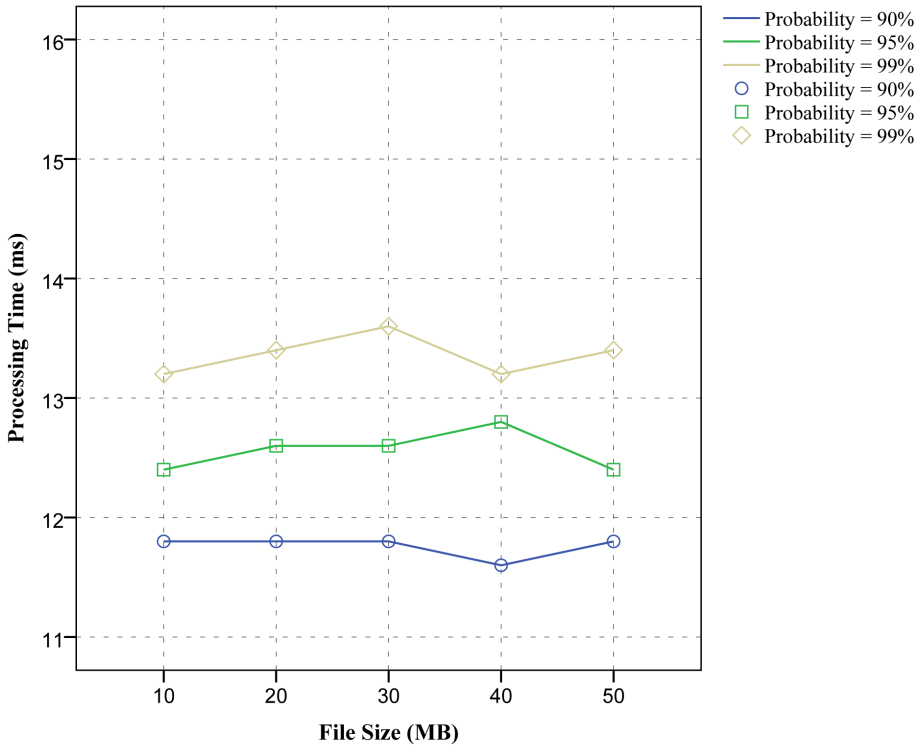


Figure 6.33: The Impact of Normal File Size on Processing Time during Verification Phase When the Signature Length is 256.

95%, and 99% probability orderly.

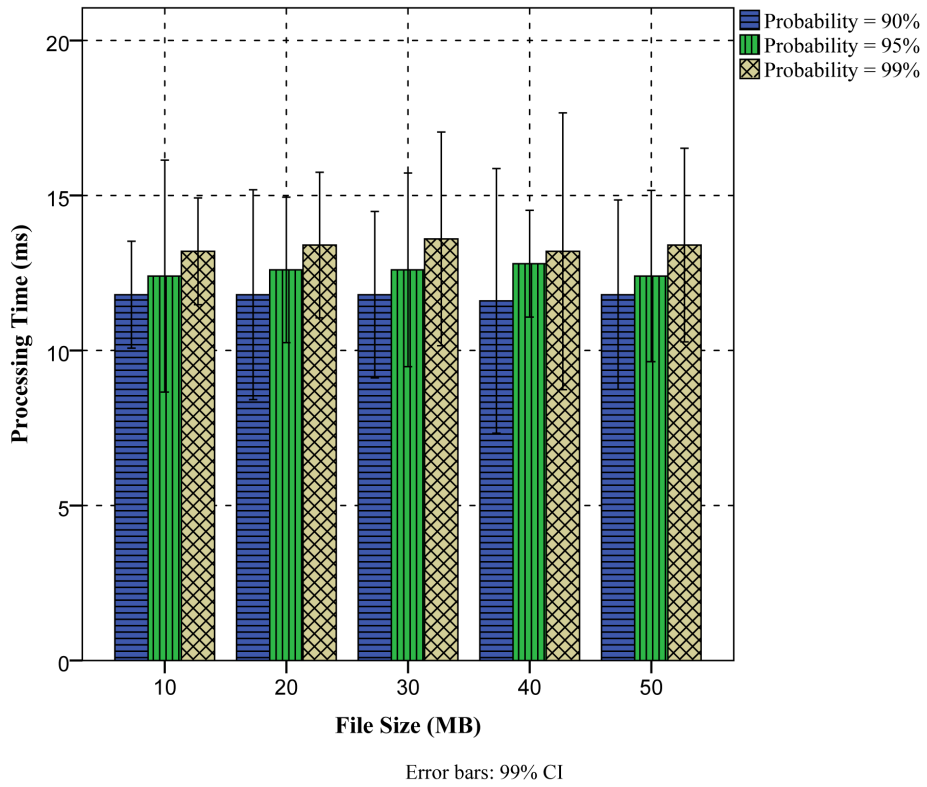


Figure 6.34: The Impact of Probability of Detection on Processing Time during Verification Phase for Normal File Size When Signature Length is 256.

Figure 6.34 proves that the processing time of the verification phase only depends on the probability of detection. For instance, when the size of file is 30 MB, the processing time is increasing from 11.80 ms to 13.60 ms by increasing the rate of detection from 90% to 99%.

The comparison of the processing time for different probability of detection during the verification phase is shown in Figure 6.35. It can be seen that there are the direct relationships between the processing time and the probability of detection. As a result, by increasing the probability of detection from 90% to 99%, the rate of processing time is also increasing. This is because the auditor must use more blocks of the file (on the basis of the number of blocks in the challenge message) to verify the integrity of the outsourced file. On the other hand, the length of signature and the processing time has the inverse relationship due to decreasing the number of sectors for computing the signature. Therefore, the minimum processing time of the verification phase is in 256 bit signature,

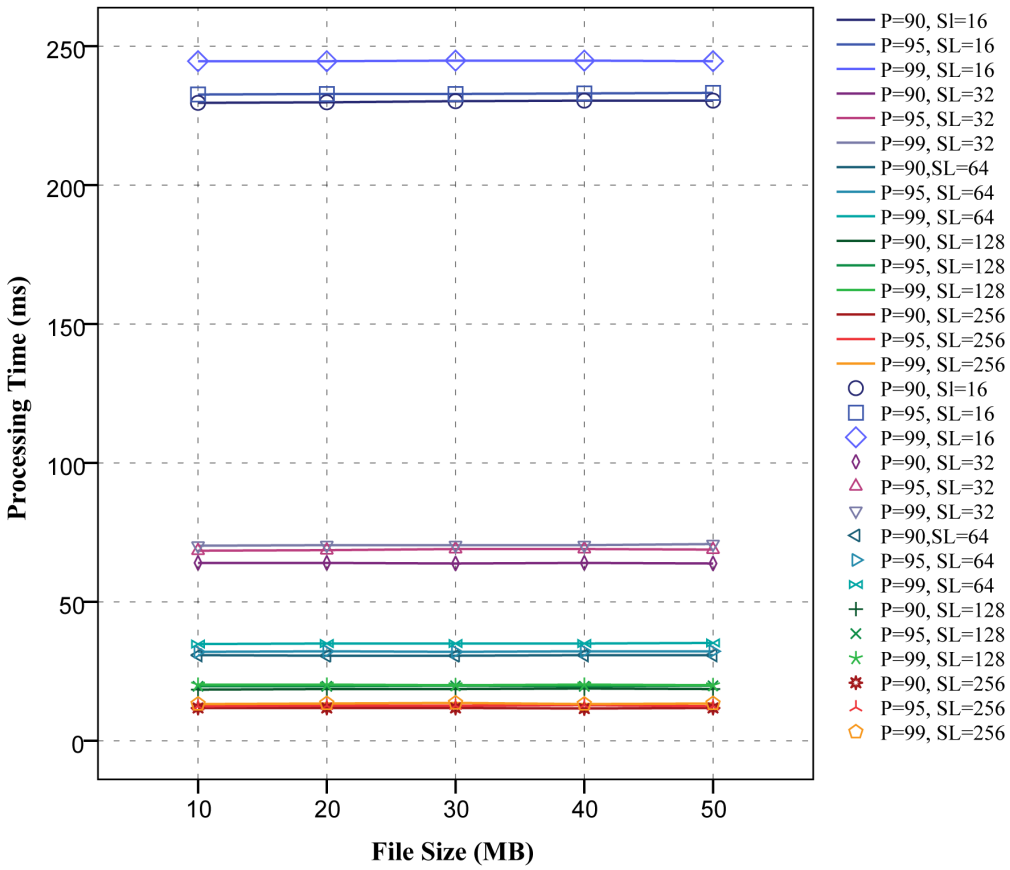


Figure 6.35: The Comparison of the Probability of Detection and Signature Length with Processing Time during the Verification Phase.

and the maximum cost is in 16 bits. Finally, the graph demonstrates that the processing time is independent of the size of file.

To ensure that the size of file has not a significant effect on the processing time, an experiment is conducted to check the integrity of large-scale file size (between 1GB – 10 GB). Figure 6.36 analyzes the impact of large-scale files on the processing time of verification phase with 256 bit signature. It is easily perceived when the probability of detection is 90%, 95%, and 99%, the processing time is in the range of 12.20 ms to 12.80 ms, 13 ms to 13.40 ms, and 15.80 ms to 16 ms respectively.

The effect of probability of detection on the processing time is also evaluated in Figure 6.37 (When the confidence interval is 99%). The graph shows there is a direct relation between the processing time and such probability in large-scale file size (as same as the normal file). For instance, the processing time of 5 GB file fluctuates between

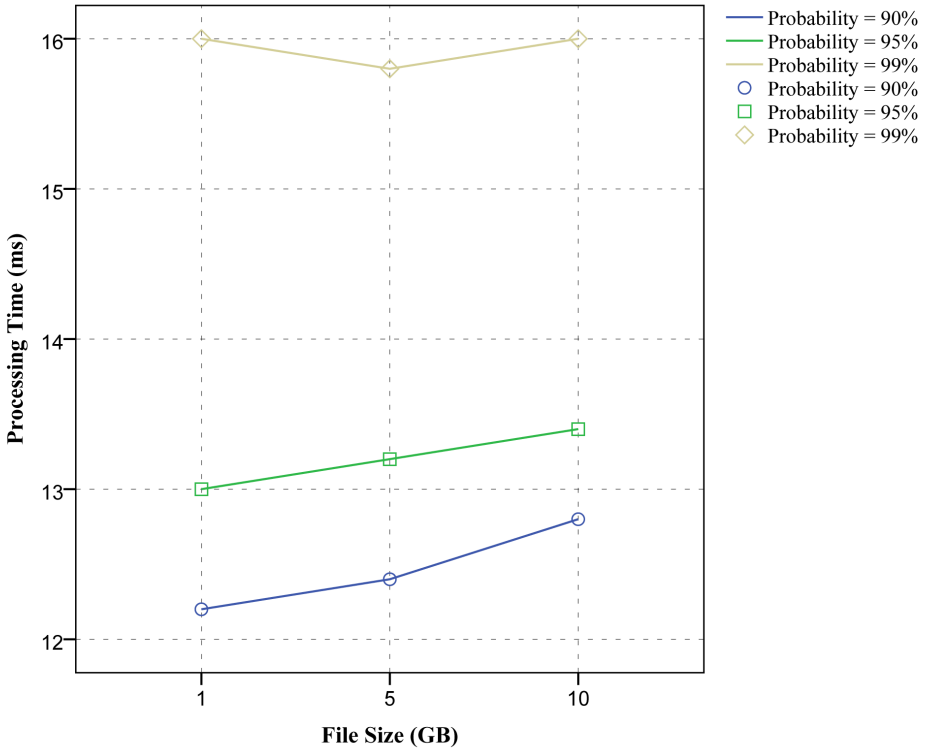


Figure 6.36: The Impact of Large Scale Files on Processing Time during Verification Phase When Signature Length is 256.

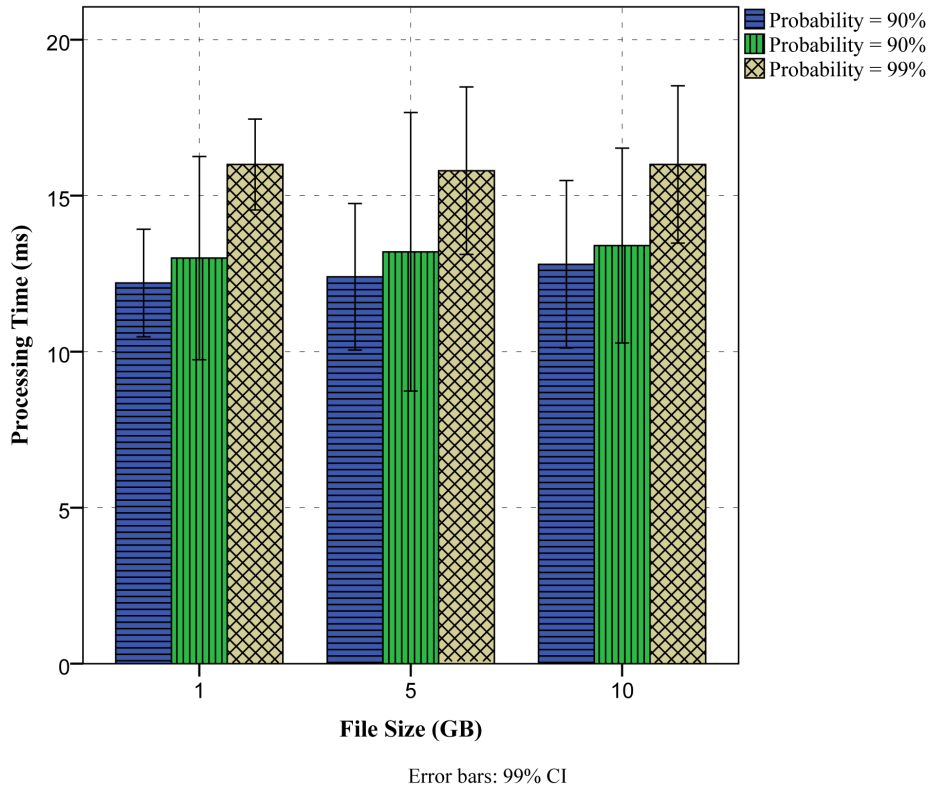


Figure 6.37: The Impact of Probability of Detection on Processing Time during Verification Phase for Large Scale Files When Signature Length is 256.

12.40 ms and 15.80 ms on the basis of the rate of detection.

6.5 Performance Analysis of Communication Cost

As aforementioned in Section 4.2.1 of Chapter 4, the proposed data auditing scheme consists of four phases: setup, challenge, response, and verification phase. There are three types of communication costs in this scheme, such as (1) communication cost of setup phase, (2) communication cost of challenge phase, and (3) communication cost of response phase. In the rest of this section, the evaluation results of these costs are analyzed.

6.5.1 Results of communication cost in setup phase

After dividing the input file into *m* blocks and computing the tag for each of them, the data owner needs to outsource the data blocks and tags to the server. The communication overhead of setup phase consists of the cost of transferring such blocks and tags, which equal to summation of all blocks, tags, and corresponding auxiliary data into the cloud. It is important to mention that such cost effects on the data owner only one time because the data owner sends the data to the cloud only one time.

The communication cost of setup phase is evaluated for two types of files: normal file size (from 10 MB to 50 MB) and large-scale files between 1GB and 10 GB. The main goal of conducting this experiment is to show the effect of signature length on the communication cost of such phase. However, to show the real overhead of the proposed method, the size of the original file is excluded from communication cost of the setup phase during this implementation.

Figure 6.38 shows the communication cost of outsourcing 10 MB file into the cloud when the signature length is from 16 b to 256 b. When the signature length is 16 b, the real cost of setup phase (by excluding the size of input file) is around 0.009766 MB. The communication cost rises to 0.019531 MB by increasing the signature length to 32 b. The maximum communication cost for uploading 10 MB file into the cloud is about 0.078125

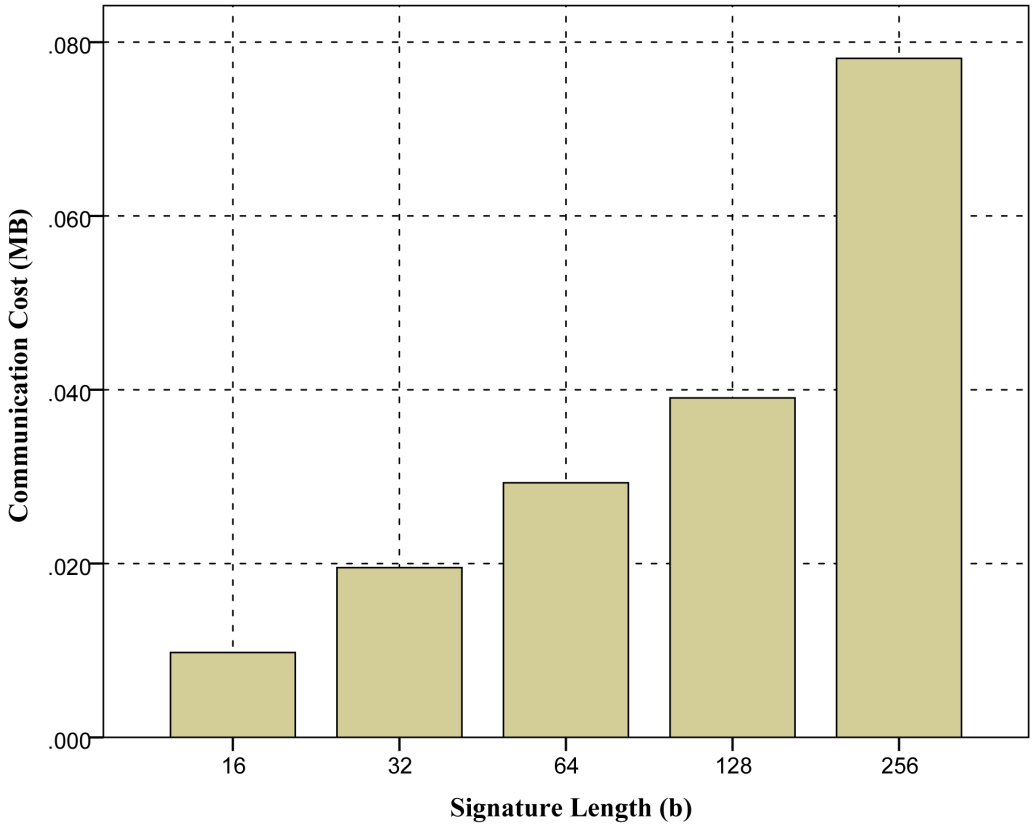


Figure 6.38: The communication Cost of Setup Phase When the File Size is 10 MB.

for 256 b signature.

The experiment is repeated for outsourcing different size of input files. For example, when the size of input file is 20 MB (Figure 6.39), 30 MB (Figure 6.40), 40 MB (Figure 6.41), and 50 MB (Figure 6.42) for different length of signature.

Therefore, by analyzing these graphs it can be seen that the communication cost of setup phase directly depends on the length of signature. In other words, by increasing the size of signature, the more communication overheads are incurred on the client.

The comparison of communication cost for different size of files (from 10 MB to 50 MB) is illustrated in Figure 6.43. The minimum communication cost is approximately about 0.00976563 MB when the signature length is 16 bits, and the size of file is 10 MB. As it is predictable, by increasing the size of file to 50 MB, the communication cost is also increasing. On the other hand, the signature length directly impacts on the communication cost. This is because by augmenting the signature length, the data owner

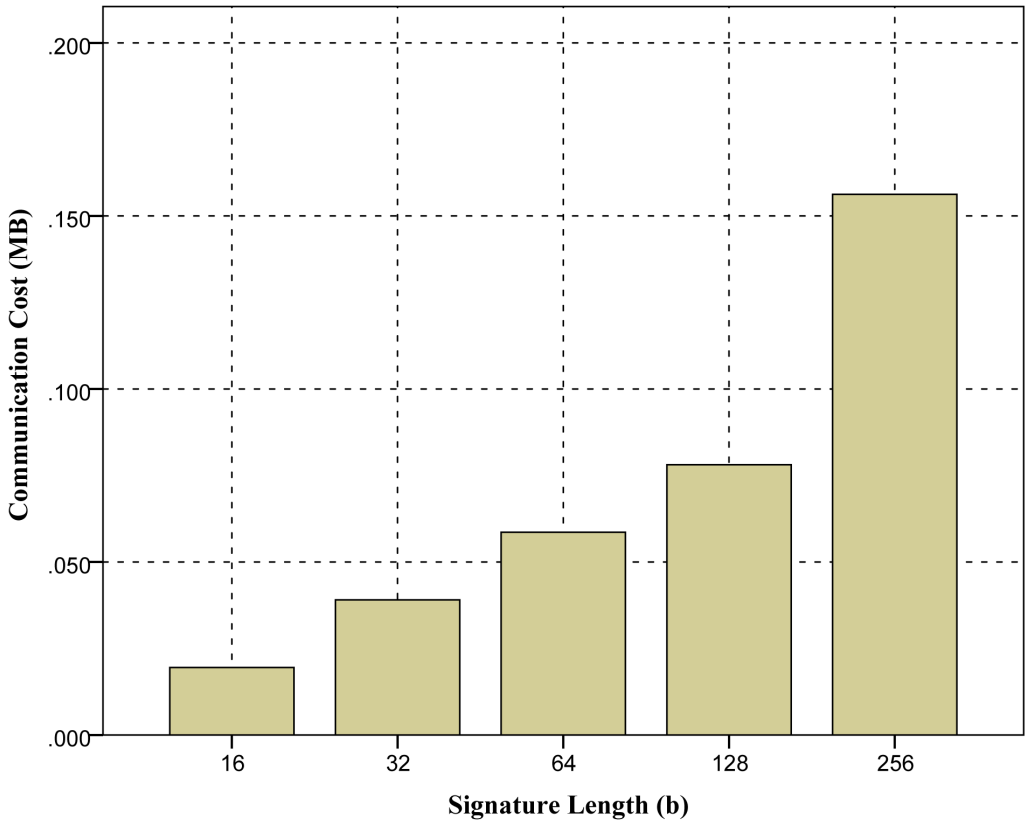


Figure 6.39: The communication Cost of Setup Phase When the File Size is 20 MB.

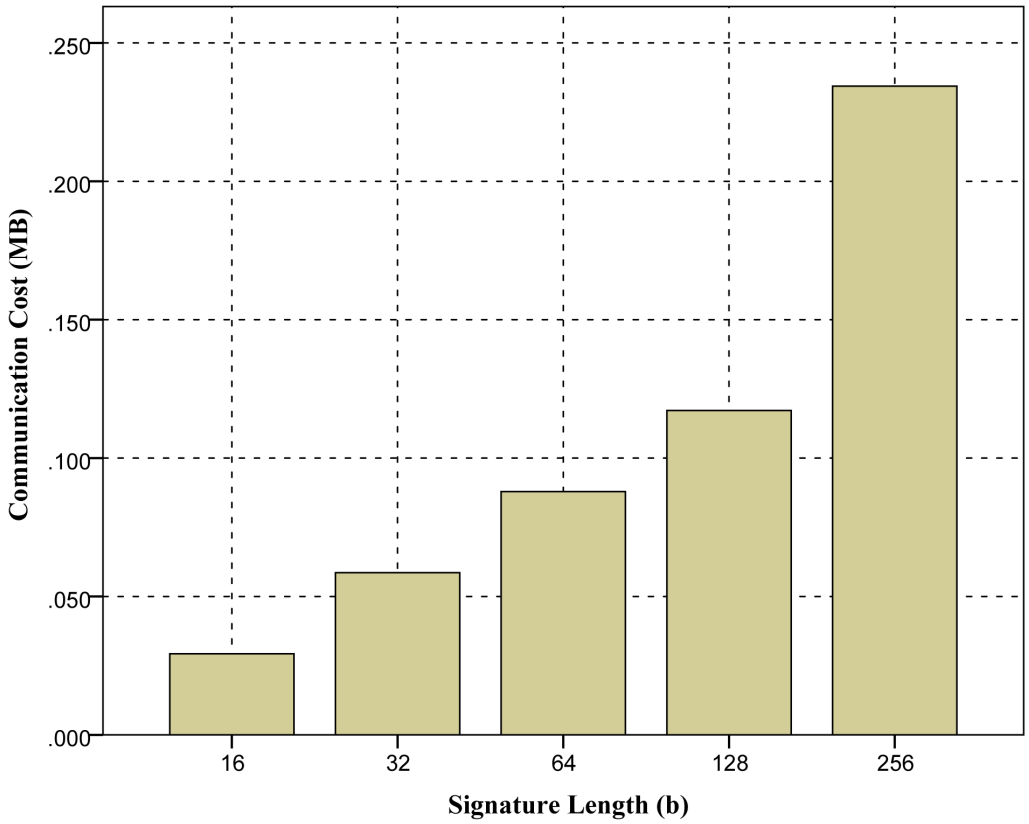


Figure 6.40: The communication Cost of Setup Phase When the File Size is 30 MB.

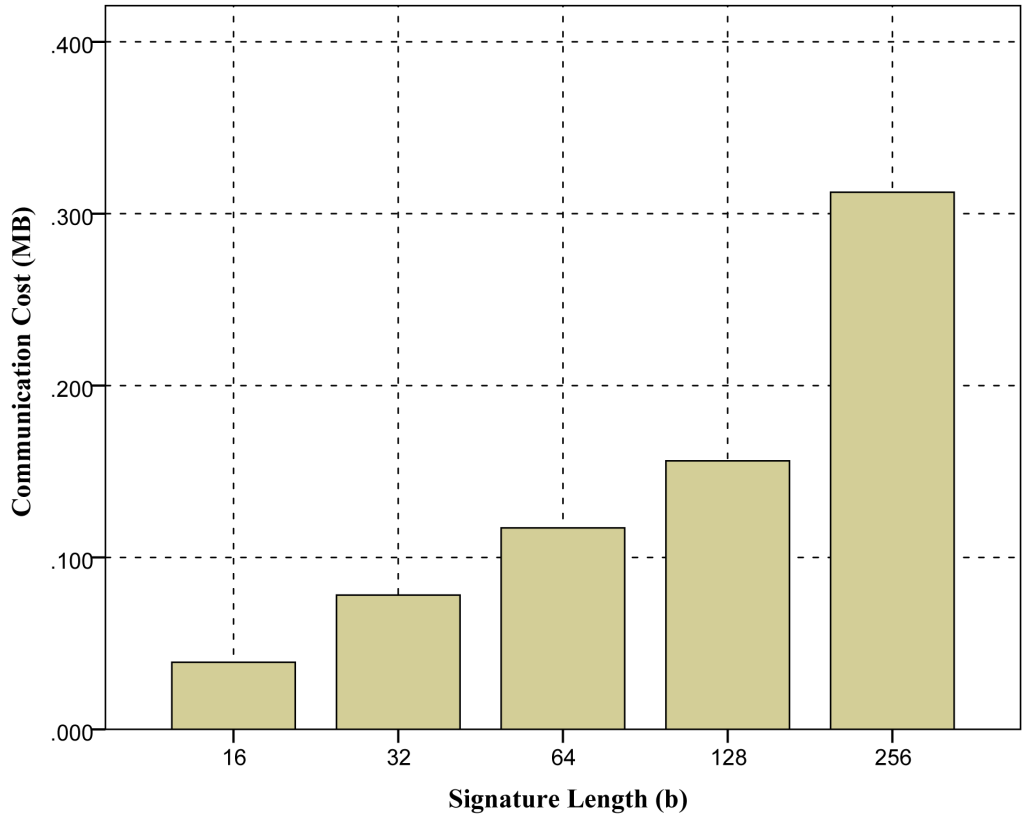


Figure 6.41: The communication Cost of Setup Phase When the File Size is 40 MB.

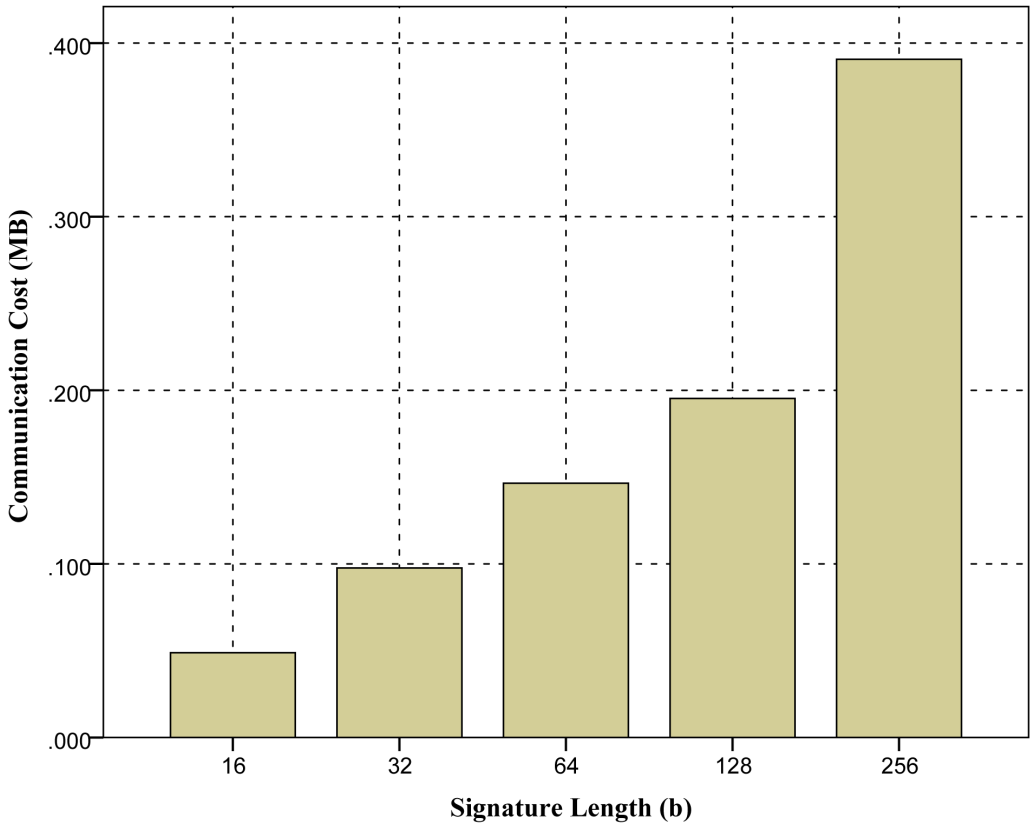


Figure 6.42: The communication Cost of Setup Phase When the File Size is 50 MB.

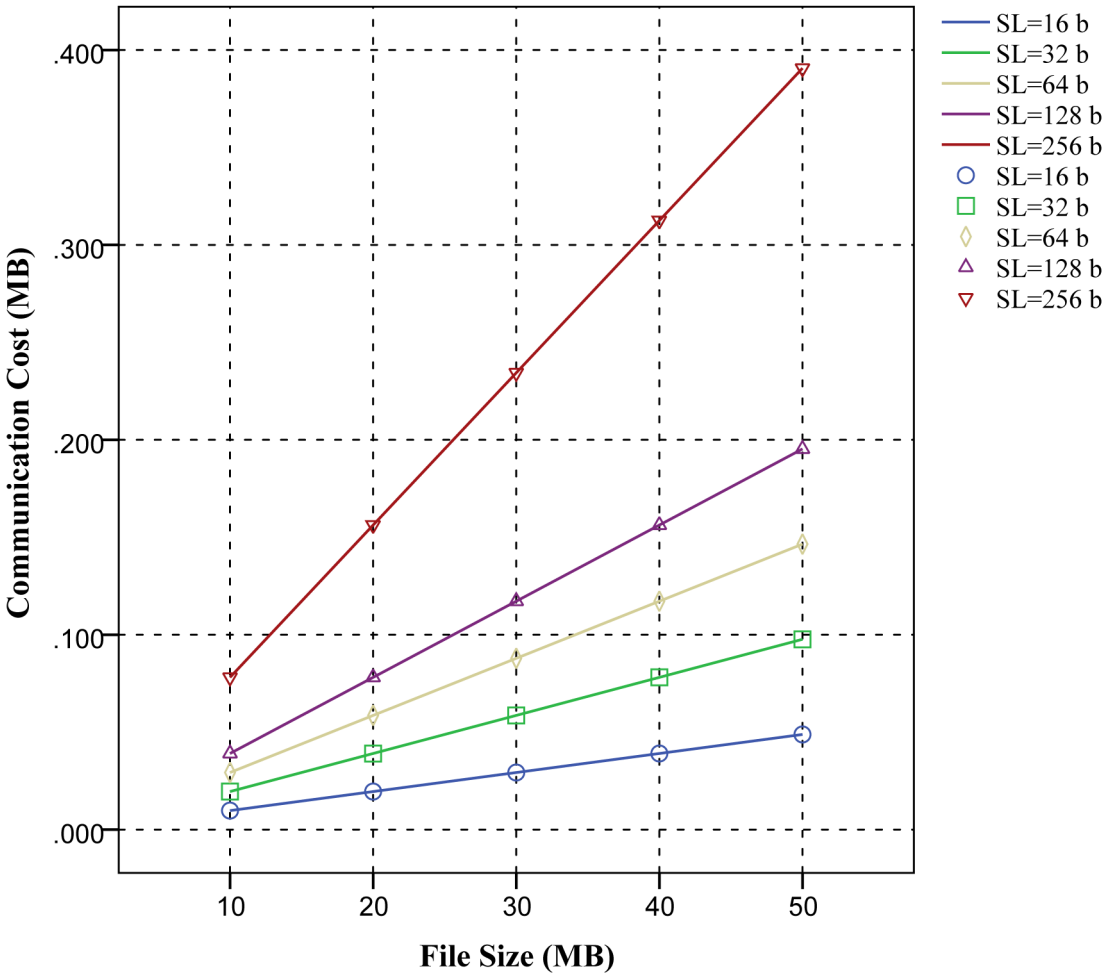


Figure 6.43: The Comparison of Communication Cost in Setup Phase for Normal File.

compels to compute and send a bigger tag for each block to the server, which is the main cause of such enhancement. The maximum communication cost belongs to a file with 50 MB size and 256 b signature length.

The effect of large-scale file on communication overhead is also checked during the setup phase for the large scale file range from 1 GB to 10 GB file size. It should be noted that we only consider the additional communication cost on the following figure to show the effect of the proposed scheme on such cost.

Figure 6.44 shows the communication cost of transferring block tags and considering auxiliary data into the cloud server during the setup phase. When the signature length is 32 b, the additional communication cost is around 0.001953 GB. By increasing the signature length to 256 b, the cost is also ascending to the maximum value (0.015625

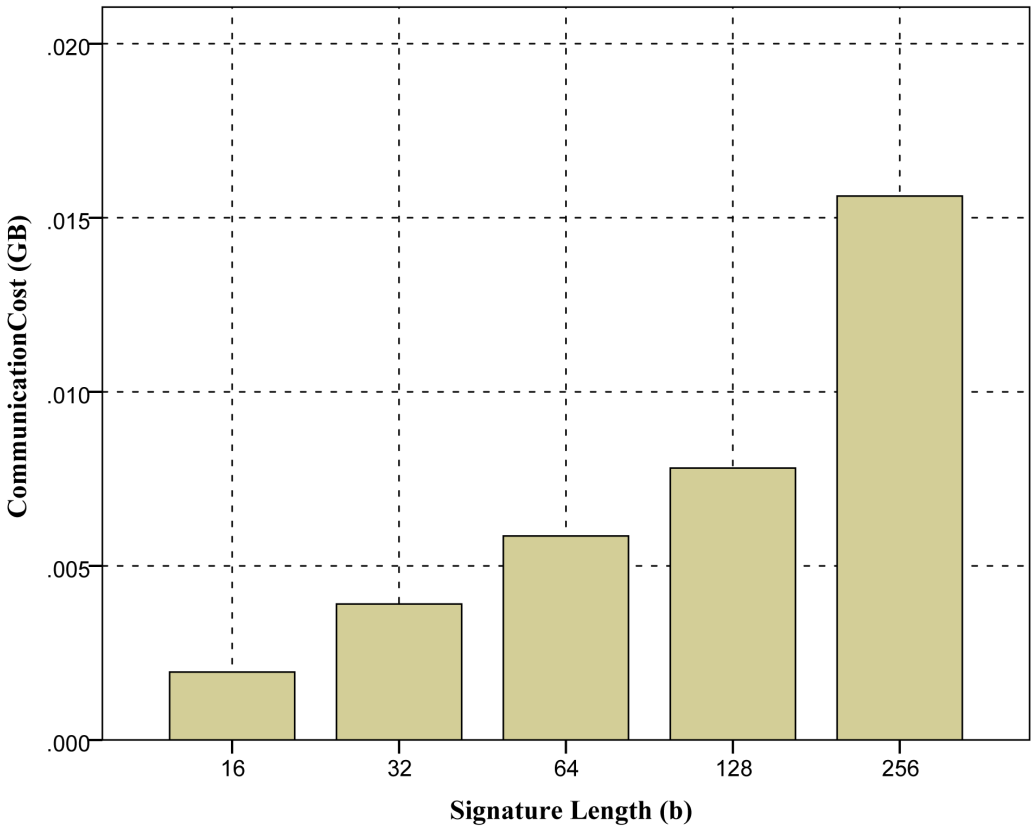


Figure 6.44: The communication Cost of Setup Phase When the File Size is 2 GB.

GB).

When the size of input file reaches 4 GB, a growing trend of communication cost can be identified explicitly. The minimum cost of setup phase of 4 GB file is about 0.0039062 when the signature length is 16 b in Figure 6.45. This cost rises to 0.031250 for 256 b signature length. Such procedure is repeated for different file size, such as 6 GB (Figure 6.46), 8 GB (Figure 6.47), and 10 FB (Figure 6.48) while the range of signature is from 16 b to 256 b.

Finally, the effect of large-scale files (1-10 GB) is evaluated on the communication overhead during the setup phase by using Figure 6.49. It is easily understood that the enhancement of the file size has a rising trend with the communication overhead in the setup phase. In addition, the signature length and communication cost has a direct relation so that minimum communication cost belongs to 16 b signature, and the maximum occurs when the length of signature is 256 bits.

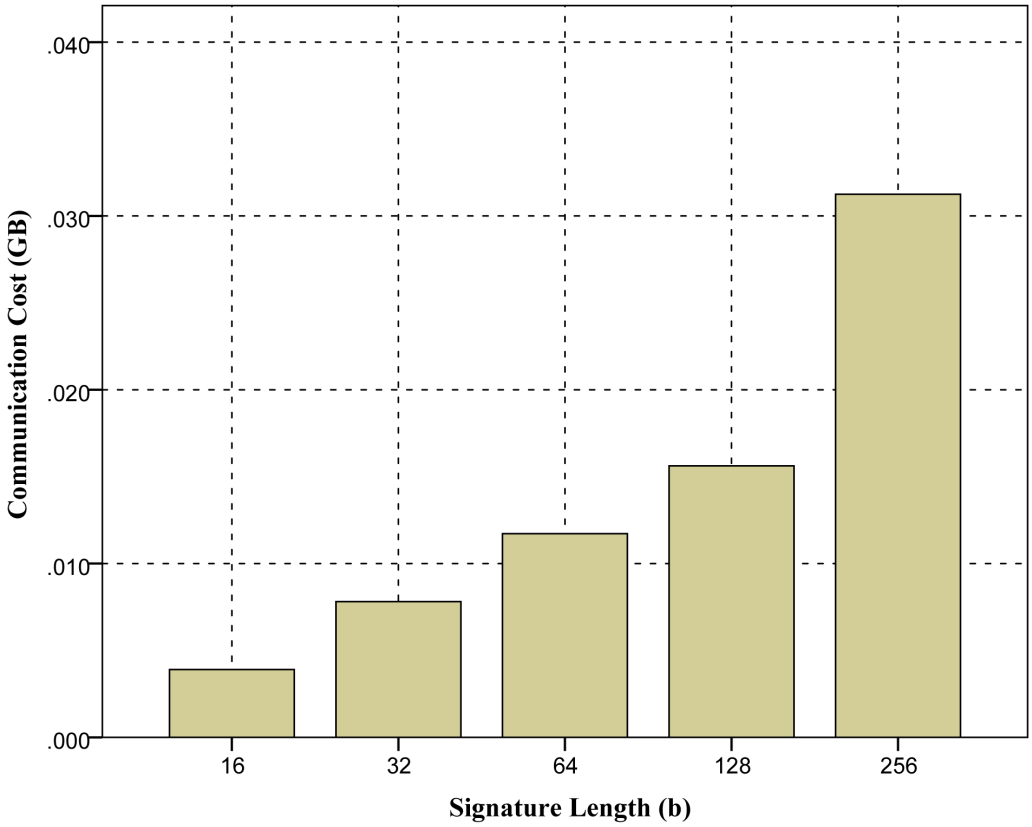


Figure 6.45: The communication Cost of Setup Phase When the File Size is 4 GB.

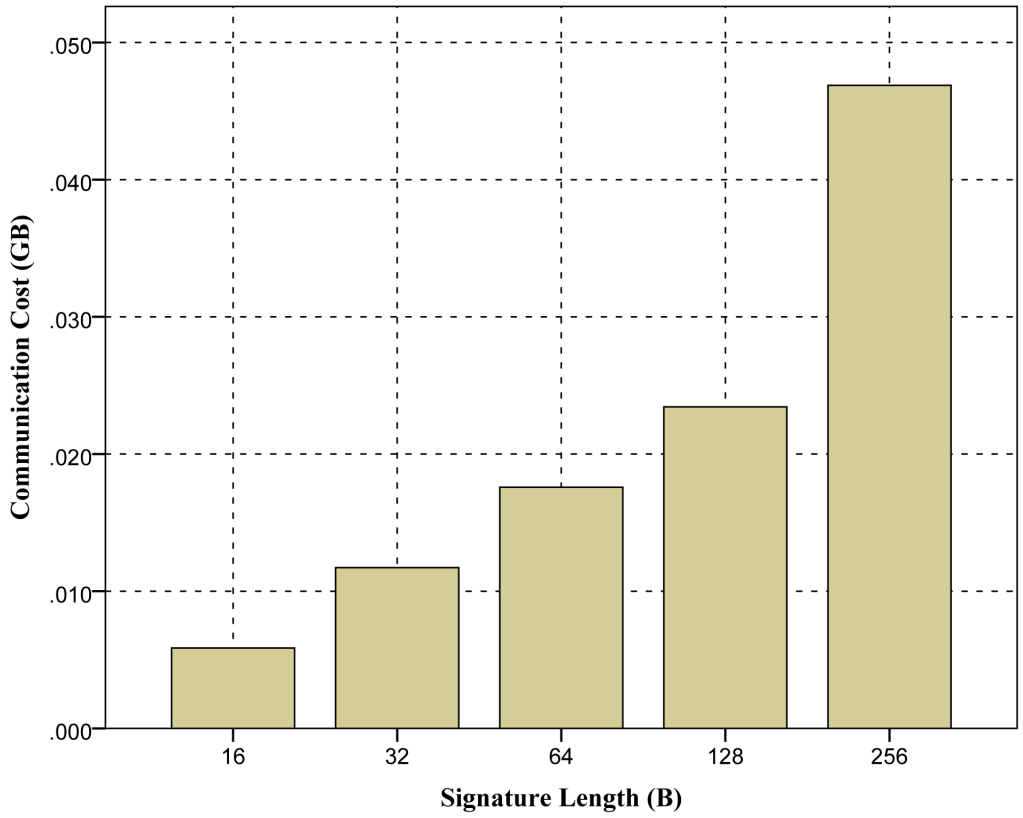


Figure 6.46: The communication Cost of Setup Phase When the File Size is 6 GB.

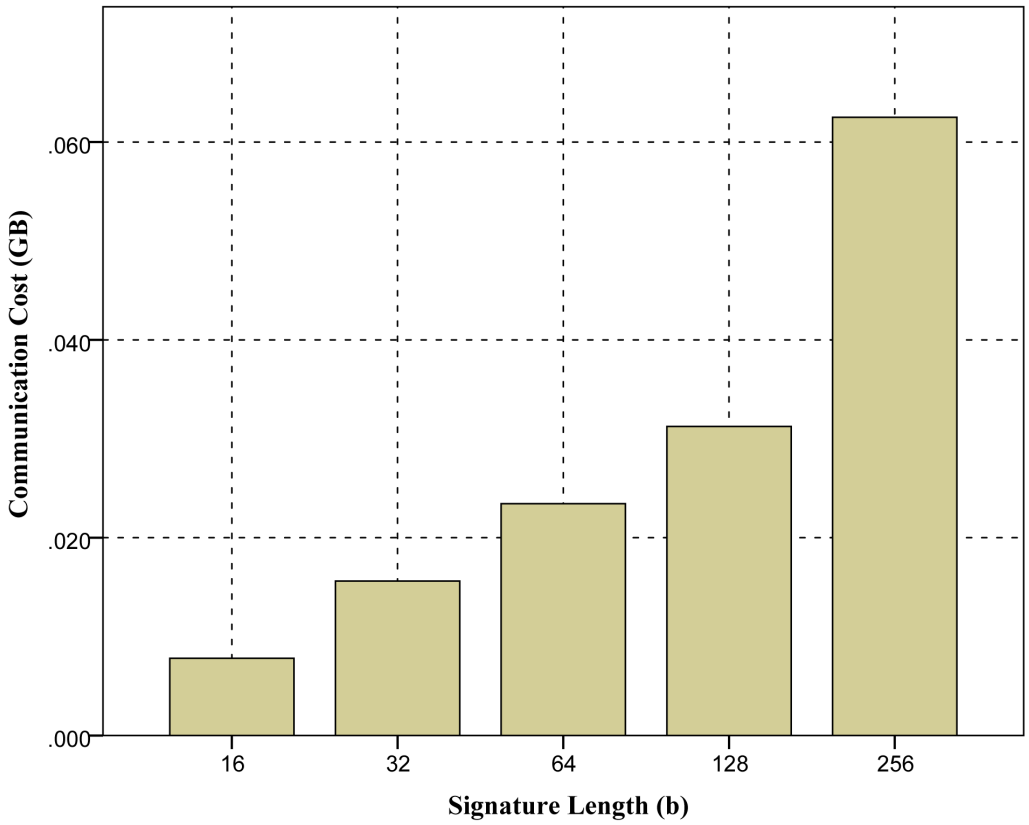


Figure 6.47: The communication Cost of Setup Phase When the File Size is 8 GB.

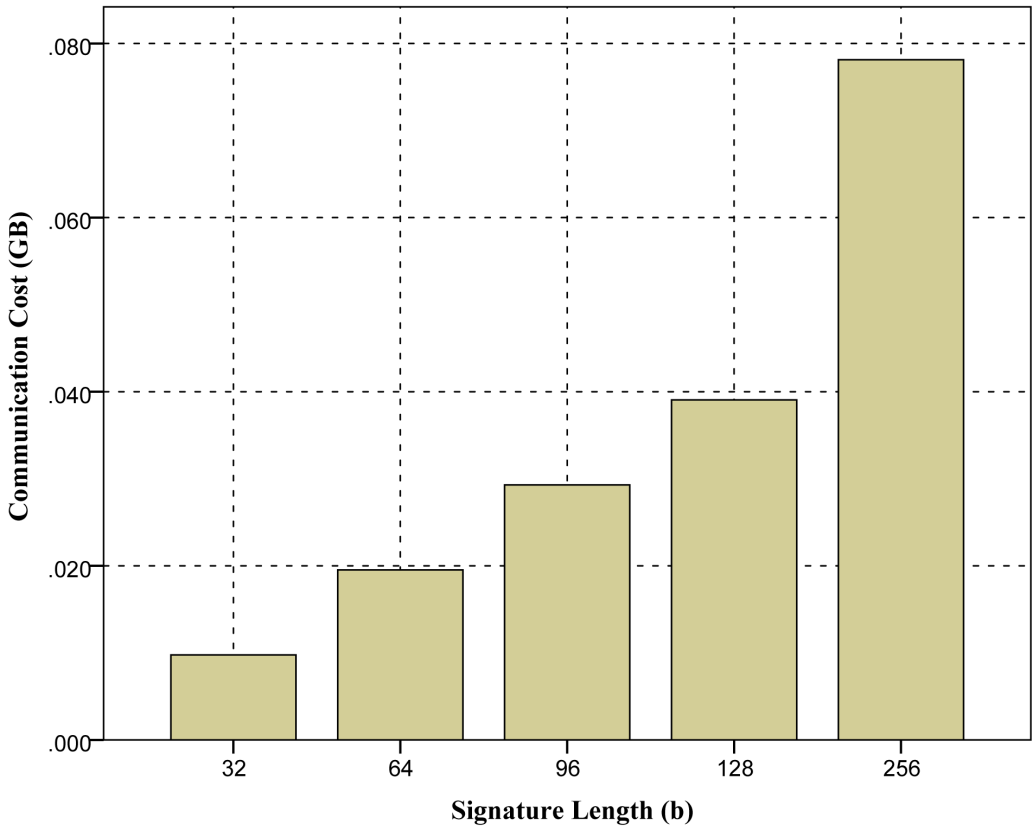


Figure 6.48: The communication Cost of Setup Phase When the File Size is 10 GB.

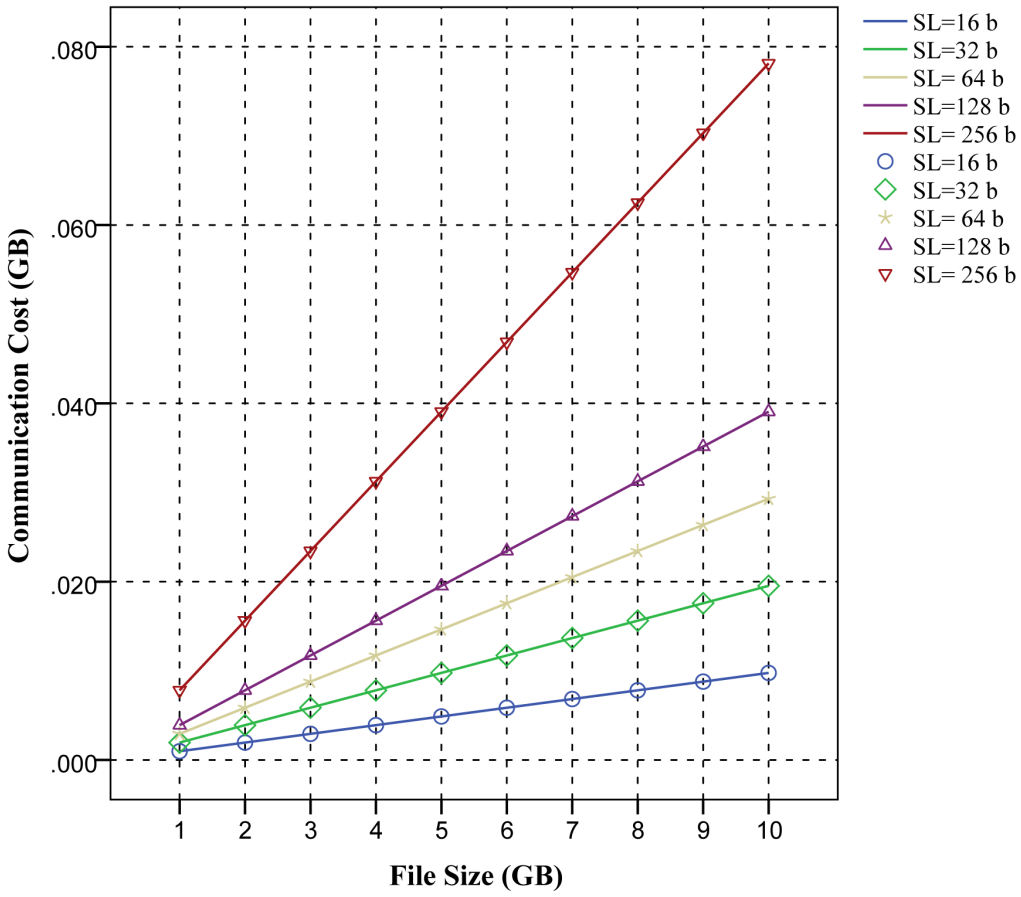


Figure 6.49: The Comparison of Communication Cost in Setup Phase for Large Scale Files.

6.5.2 Results of communication cost in Challenge phase

The communication cost of challenge phase indicates the amount of data that is transferred to the server during the challenge phase. Since the challenge message only includes a random set of indices of blocks, the communication cost of challenge phase is independent from size of input file and length of signature. In other words, the amount of such data only depends on the number of selected blocks during the challenge phase that is called probability of detection.

Figure 6.50 displays the communication cost of challenge phase for normal and large-scale files. It can be understood that the rate of communication cost in challenge phase for 90% probability is around 0.9 KB. By increasing the probability of detection to 95%, the communication cost reach 1.17 KB. Finally, the communication cost of chal-

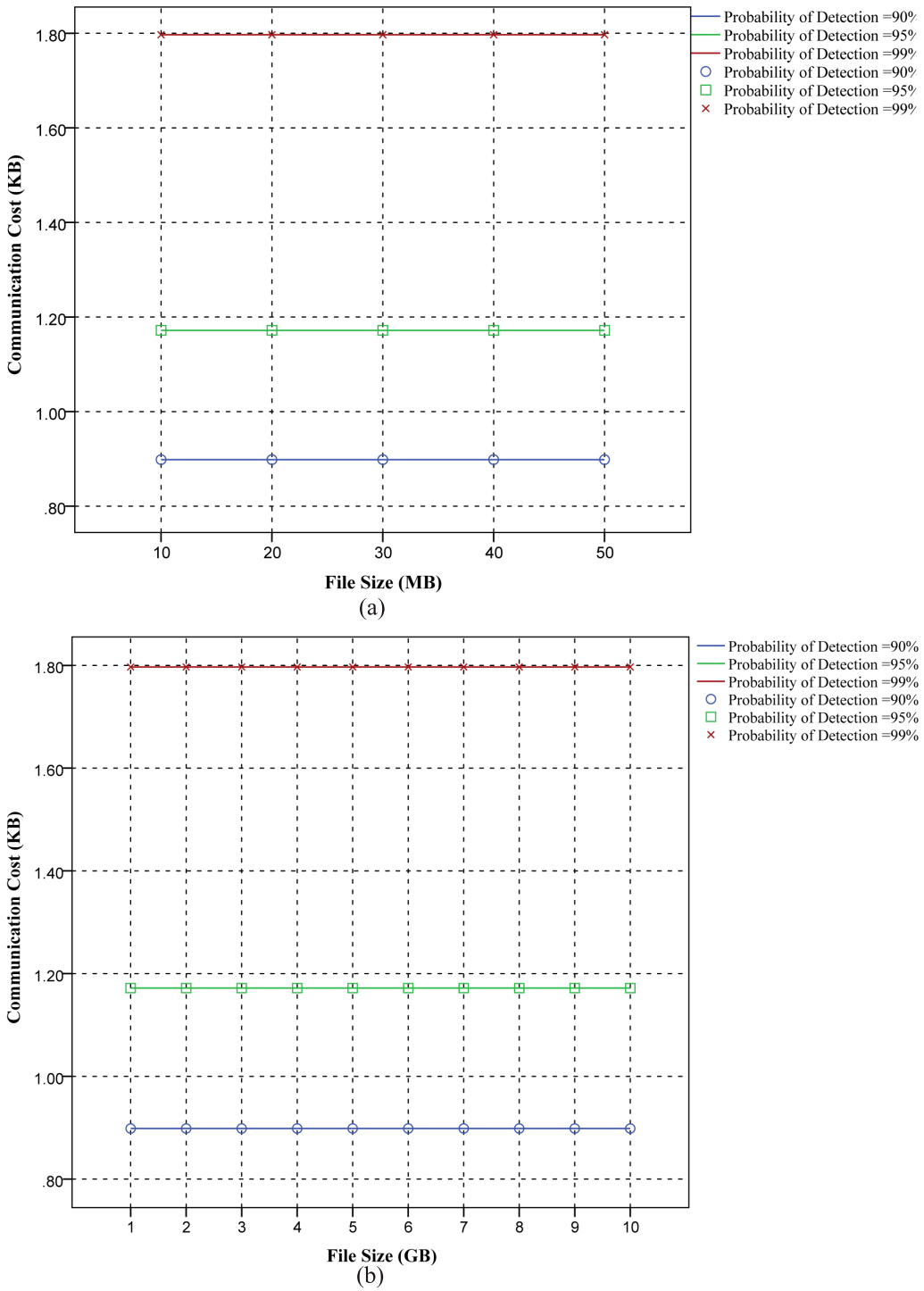


Figure 6.50: The Comparison of Communication Cost in Challenge Phase for (a) Normal File Size, (b) Large Scale Files.

lenge phase for 99%probability is about 1.80 KB.

6.5.3 Results of communication cost in Response phase

The communication cost of response phase indicates the amount of transferred data from prover to the auditor in the response phase.

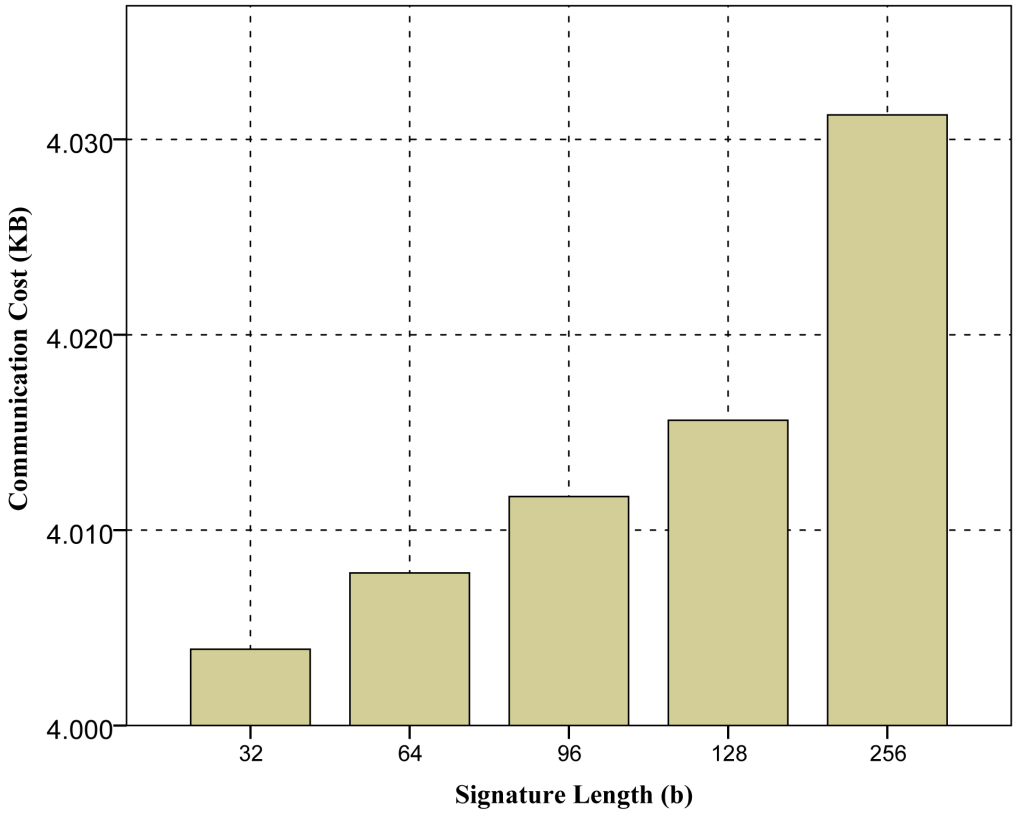
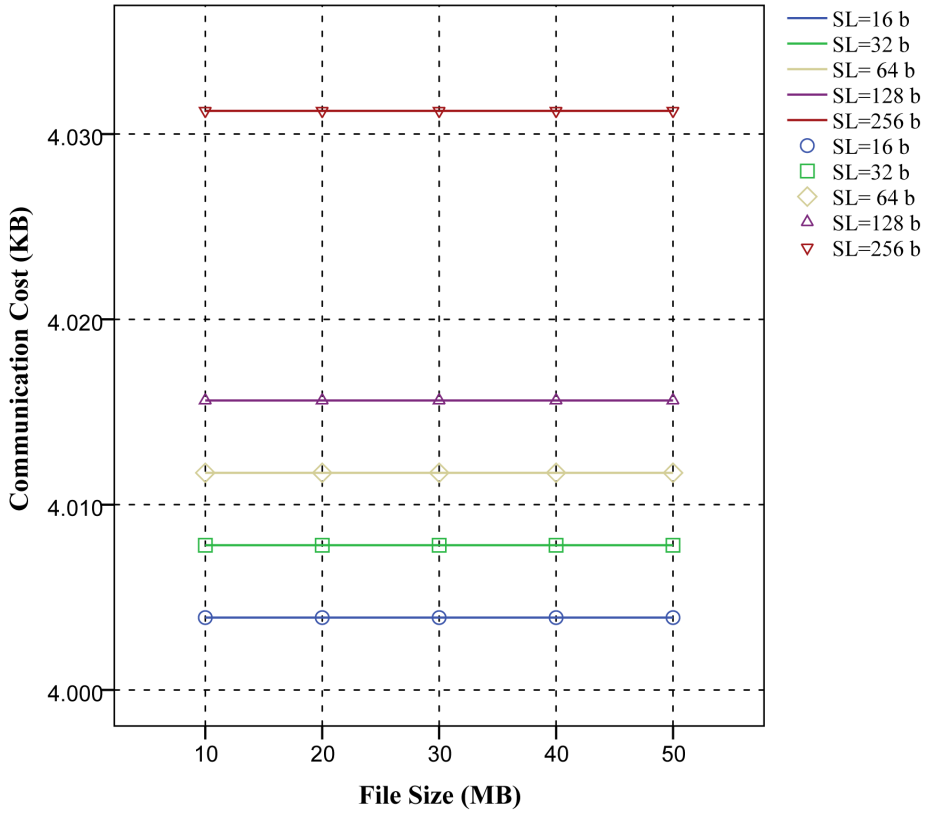


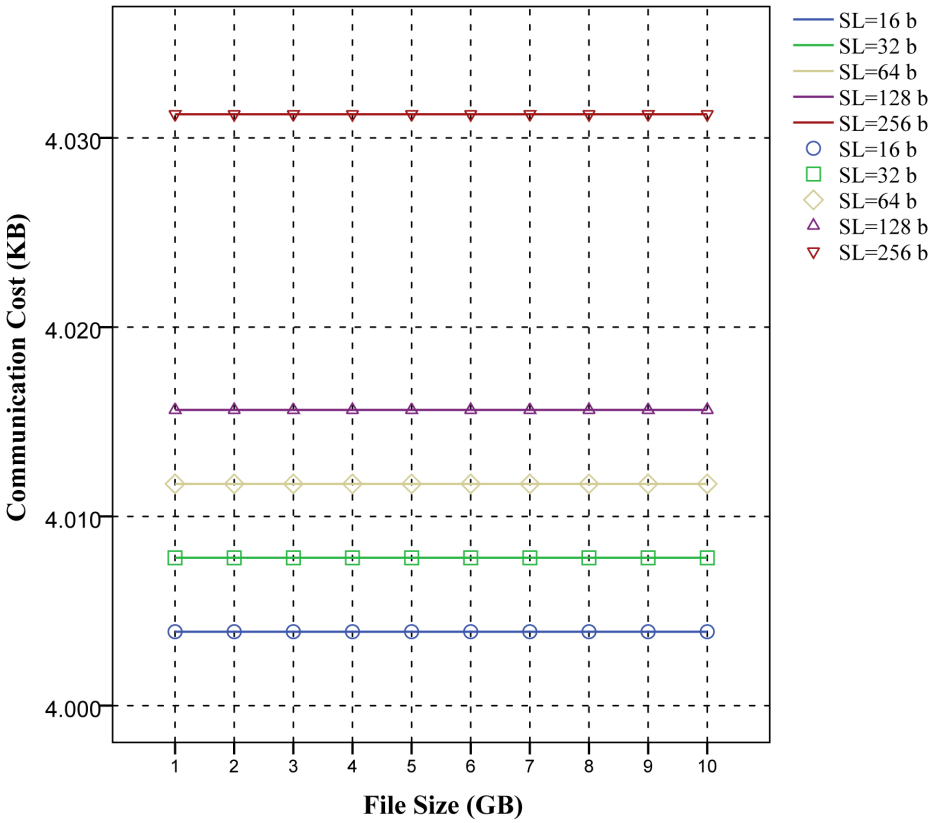
Figure 6.51: The Communication Cost of Response When the Signature is from 16 b to 256 b for any File Size

Since for any number of data blocks as a challenge message the prover compels to generate the fixed length of message as a response (consists of a linear combination of the blocks (σ) and the aggregate authenticator tags (μ)), the communication cost of response phase is independent of the probability of detection and the file size. The effect of signature length on the communication cost of the response phase is illustrated on Figure 6.51. It is easily perceived that by increasing the length of signature from 16 b to 256 b, the communication cost is increasing slightly from 4.003906 KB to 4.031250 KB. Moreover, the communication cost of response phase is independent of the size of file.

Figure 6.52 shows a comparison of the communication cost during the response phase for normal file size (a) and large-scale files (b). It can be seen that the communication cost of the large-scale files (1 GB-10 GB) has the same attitude as the rate of communication cost for normal size (10 MB-50 MB). In other words, the communication



(a)



(b)

Figure 6.52: The Comparison of Communication Cost in Response Phase for (a) Normal File Size, (b) Large Scale Files

cost of the response phase is independent from the size of file.

6.6 Performance Analysis of D&CT Divisions

The D&CT has an important role in the proposed method to support dynamic data update effectively. As mentioned in section 4.3, to decrease the processing time of data updating, the number of divisions of D&CT (k) have to be set on the basis of the number of data blocks (m). In this section, two scenarios are defined to evaluate the effect of division feature of D&CT on the processing time of data update on the auditor, as follows.

The first scenario clarifies the importance of division technique that is used in the D&CT. To achieve this goal; we conduct the experiments for updating several outsourced files (F) with length from 1 GB to 100 GB, including 125,000 to 1,125,000 data blocks. The number of division are from 10 to 353 and the number of insert or delete operations is 1000. The position of the blocks that are selected randomly, keep constant for all numbers of divisions.

Figure 6.53 illustrates that when the size of outsourced files or the number of data blocks are increased; remarkable processing time is incurred on the auditor. For example, when the number of divisions is 10, the computation time for updating 1000 blocks of a file with size 1 GB is about 0.218 s. By increasing the file size to 100 GB, the overhead rises to 10.811 s. The result also shows that the computation overhead falls down dramatically by increasing the number of divisions as such an overhead for updating 1000 blocks of a file with size 100 GB is only about 0.296. When the number of divisions reach 353, the processing time is in the range of 0.140 s to 0.296 s for 1 GB to 100 GB files. Therefore, the proposed data structure has a significant role in decreasing the computation overhead when the data owner needs to upload a huge file or update the outsourced file frequently.

In the last scenario, the effect of the number of divisions on the number of update blocks is checked in terms of processing time. The size of outsourced file is 1 GB, number

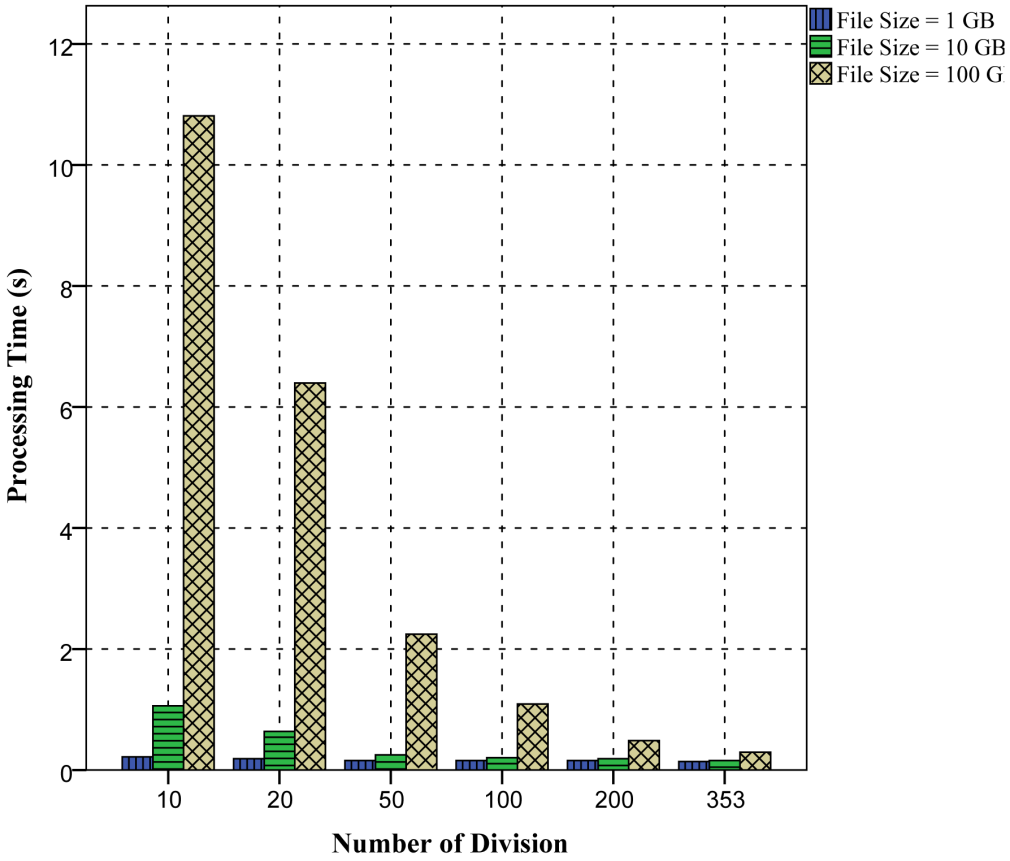


Figure 6.53: The impact of number of divisions on computation time under different file size from 1 GB to 100 GB

of update operations (insert or delete) is in the range of 100 to 3000, and the number of divisions can be 10, 20, 50, 100, 200, and 353.

As aforementioned in section 4.3 of chapter 4, the optimal number of divisions (k) is equal to $\sqrt{n}$ where n indicates the number of blocks. Figure 6.54 illustrates the effect of division on the incurred processing time on the auditor. It is easily perceived that by increasing the number of divisions (k) and approaching to the optimum number (353), the processing time on the auditor is decreasing. Moreover, when the numbers of updated blocks are increasing, the considerable computation overhead is incurred on the auditor. One of the best ways to reduce such a computation overhead is to use the optimum number of divisions based on the file size. For example, when the number of divisions is $k = 100$, the computation time of inserting 1000 blocks is around 0.218. Upon increasing the number of divisions to 353, the computation times fall down to 0.140 s.

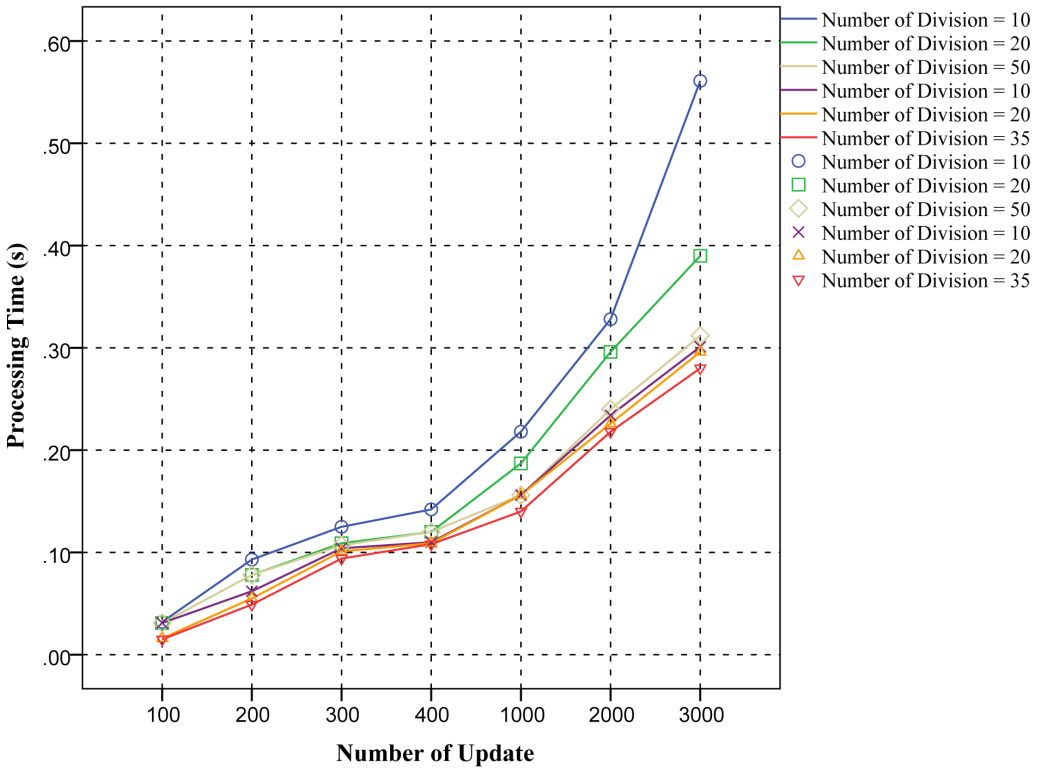


Figure 6.54: The Impact of Number of Divisions on Processing Time under Different Number of Update Blocks

The main reason of conducting the first part of experiment in section 6.4, Section 6.5, and section 6.6 is to identify the effect of input parameters, such as file size, length of signature, probability of detection, and number of divisions on the computation and communication cost of the DRDA method. As it is mentioned in the section 6.4.4 of chapter 6 and section 3.2 of chapter 3, the processing time of the verification phase is the most important cost of the remote data auditing schemes. It also shows that when the length of signature is equal to 256 b, minimum computation overhead is incurred on the auditor. The result of section 6.6 also shows that the number of divisions in the D&CT data structure has a direct impact on the processing time of dynamic data update.

6.7 Summary of Findings and Discussions

This section provides a discussion about the implementation results of the proposed scheme and compares them with state-of-the-art schemes to proof the efficiency of the method.

To check whether there are any significant differences between the means of experimented methods, a statistical significance analysis is also conducted in this part. Since the number of pair-wise comparison is more than two methods, it requires applying the one-way analysis of variance (ANOVA) as a way to compare the results. However, the one-way ANOVA is unable to show which two particular methods are significantly different from each other; it only indicates that at least two methods were different. Therefore, Games-Howell as a post-hoc test was leveraged to perform a multiple comparisons.

6.7.1 Analysis the effect of implementation parameters on performance

Processing time and communication cost are two important measures that are used to evaluate the performance of the proposed method. Table 6.1 shows the effect of implementation parameters such as file size, signature length (Signature), and probability of misbehavior detection (Detection) on the communication cost and processing time of the DRDA method, as follows:

(i) ***Processing Time of setup phase:*** As illustrated in Figure 6.3, the processing time of setup phase is related directly with the file size. It is because by increasing the size of file, the data owner must compute tags for more data blocks. However, growing the length of signature caused decreasing the processing time during such phase because of declining the number of divisions to compute the tags (Figure 6.9). The probability of detection parameter is unable to change the rate of the processing time of setup phase.

(ii) ***Processing Time of challenge phase:*** There is a direct relationship between the processing time of challenge phase and the probability of detection (Figure 6.13), due to the effect of this parameter on the number of blocks in the challenge phase. Moreover, augmenting the file size causes growth in the range of data blocks. Therefore, the file size has a direct relation with the processing time of challenge

phase. Finally, the length of signature is independent from the processing time of this phase.

(iii) ***Processing Time of the response phase:*** There is a direct relation between processing time and probability of detection in the response phase. This is because the proof message that includes the linear combination of the blocks (σ) and the aggregate authenticator tags (μ) is only computed on the basis of the challenge message (that was generated by using the sampling technique in the challenge phase). As a result, the size of file and signature length has not a significant effect on the processing time of response phase (Figure 6.25).

(iv) ***Processing Time of the verification phase:*** The Length of signature has an inverse relation with the processing time of the verification phase because increasing the length of signature results in decreasing the processing time of generating the algebraic signature. On the other hand, increasing the number of blocks in the challenge phase (probability of detection) directly affects the processing time (Figure 6.32). Finally, the result shows that the size of the outsourced file has a negligible effect on the processing time of verification phase.

(v) ***Communication cost of setup phase:*** The message of setup phase consists of the data blocks, considering tags, and auxiliary data. Therefore, the amount of such messages as a communication cost have a direct relation with size of blocks and signature size that has effect on size of tag. However, the probability of detection is an independent parameter in this phase (Figure 6.45).

(vi) ***Communication cost of challenge phase:*** During the challenge phase, a random of data blocks and coefficients are generated on the basis of probability of detection. Therefore, the communication cost of challenge phase depends on such probability

when the auditor uses a sampling technique.

(vii) ***Communication cost of response phase:*** The response message includes an integration of challenged blocks and considering tags. Therefore, when the size of data blocks is fixed, the size of file has no relation with the communication cost of the response messages. The length of signature is the only factor that has a direct relation with such cost (Figure 6.47).

Table 6.1: Reviewing on the Relationship between Various Parameters of the Proposed Scheme

Cost	**Phase**	**File Size**	**Signature**	**Detection**
Computation	Setup	$\triangle$	$\bigtriangledown$	$\bowtie$
	Challenge	$\triangle$	$\bowtie$	$\triangle$
	Response	$\bowtie$	$\bowtie$	$\triangle$
	Verification	$\bowtie$	$\bigtriangledown$	$\triangle$
Communication	Setup	$\triangle$	$\triangle$	$\bowtie$
	Challenge	$\bowtie$	$\bowtie$	$\triangle$
	Response	$\bowtie$	$\triangle$	$\bowtie$

Independent: $\bowtie$ Direct: $\triangle$ Inverse $\triangle$

6.7.2 Comparison of processing time of data integrity with the traditional RDA methods

This section validates the proposed scheme by comparing the processing time of DRDA method with the processing time of the traditional RDA methods in the same situation. To perform the comparison, several parameters need to be defined in Table 6.2, such as size of file, probability of detection, corruption rate, and signature length.

Table 6.2: The Comparison parameters

Parameters	Value
File Size	10 MB - 50 MB
Probability of Detection	90% - 99%
Number of Challenges	230 - 460
Block Size	8 KB
Corruption rate	0.01%
Signature Length	256 b

Figure 6.55 shows the comparison between the traditional RDA methods (PDP, and Public PDP) and the proposed method (DRDA) on the basis of the processing time of data integrity when the probability of detection is 90% and 99% respectively. It examined that the processing time of PDP method is around 250 millisecond and 400 millisecond for 90% and 99% probability. The processing time of public PDP method increased to 282 millisecond and 805 millisecond approximately for different rate of probability. The

graphs demonstrate that the processing time of DRDA method is much better than PDP and public PDP methods, which is about 12 millisecond and 14 millisecond when the probability of detection is 90% and 99%. It is because of applying the algebraic signature and the D&CT structure.

6.7.3 Comparison of processing time during dynamic data update

This section analyzes the comparison of processing time of dynamic data update for different number of update operations. The usefulness of the proposed method is evaluated in terms of dynamic data update when the size of file is large-scale (1 GB – 10 GB). The probability of detection is 99%, size of data block is 8 KB, the number of divisions is 350, and the length of signature is 256.

Figure 6.56 shows the comparison of the processing time of dynamic data update in the DRDA and public PDP method for large-scale files when the number of update operations is 2. It is examined that processing time of dynamic data update reduces significantly in the DRDA method due to applying D&CT data structure. Experimental result indicates that the processing time of dynamic data update in public PDP is around 1.65 second, while the processing time of the proposed method reduces to 0.03 second.

Figure 6.57 shows the comparison of processing time of dynamic data update in the DRDA and public PDP method when the number of update operations is 4. The experimental result demonstrates that processing time of dynamic update is around 3.31 second when the large-scale outsourced file is updated 4 times. The processing time of the DRDA method significantly reduces to 0.06 second due to the structure of the D&CT.

The comparison of processing time of dynamic update in DRDA and public PDP method when the number of update is 6 is illustrated in Figure 6.58. It is examined that processing time of dynamic update in the proposed method decreases significantly. For instance, the processing time of public PDP is approximately 4.96 second while the

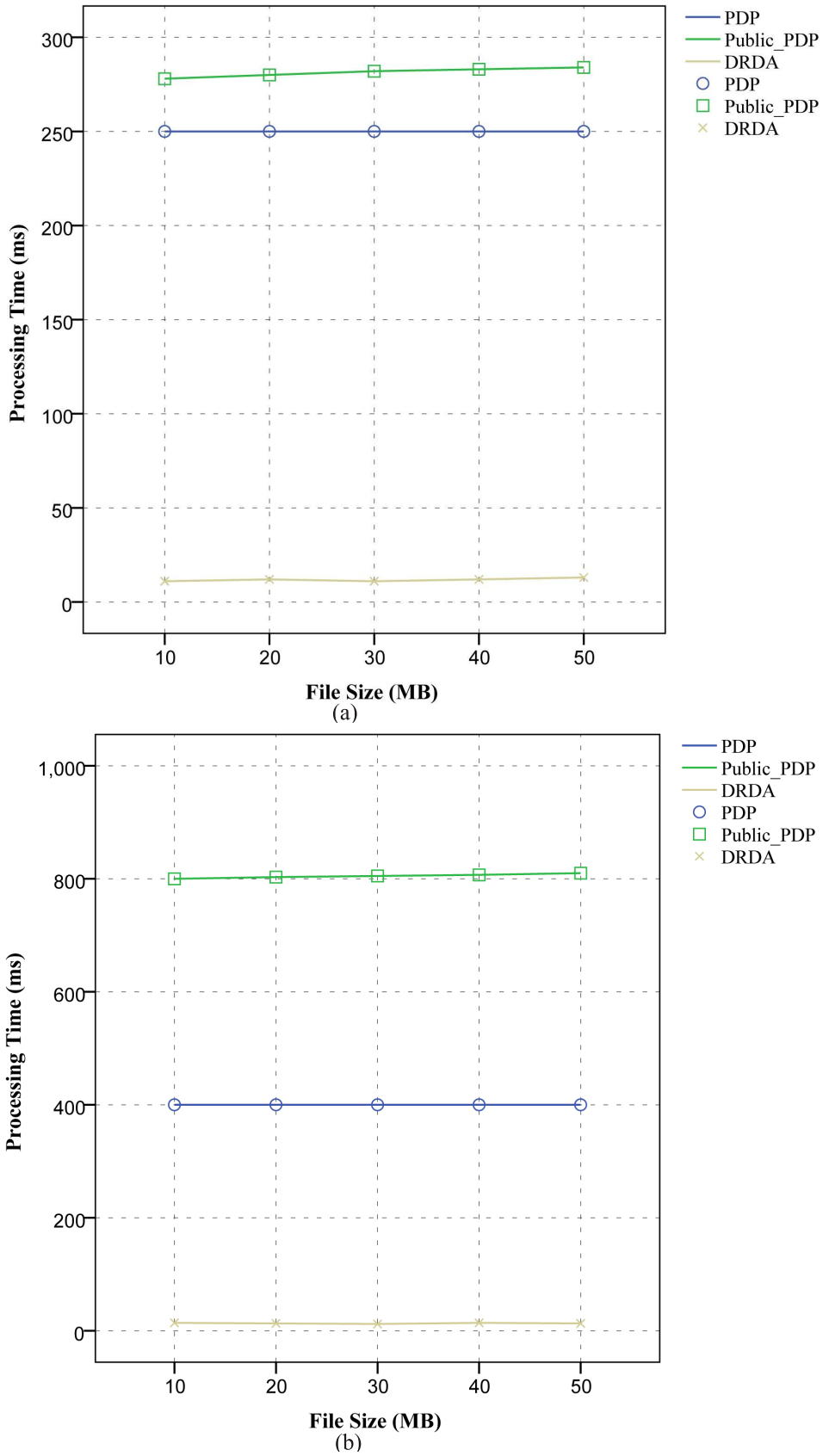


Figure 6.55: The Comparison of the Processing Time between the Traditional RDA method and the Proposed Method for (a) 90% Probability of Detection, and (b) 99% Probability of Detection

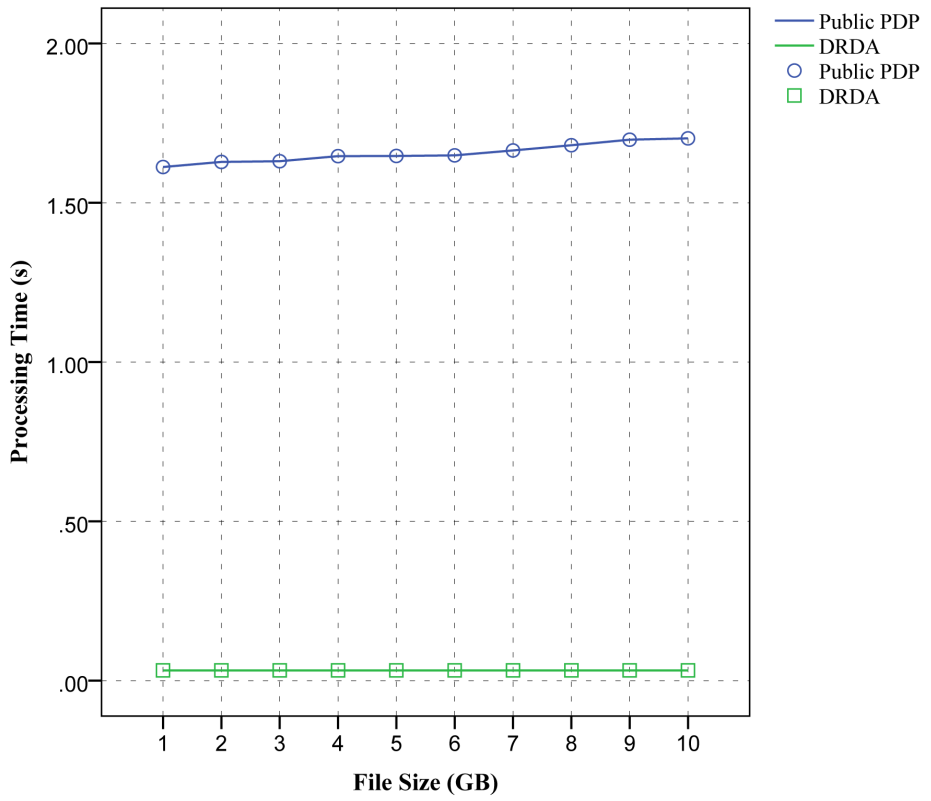


Figure 6.56: Comparison of processing time of Data Update when the number of updates is 2

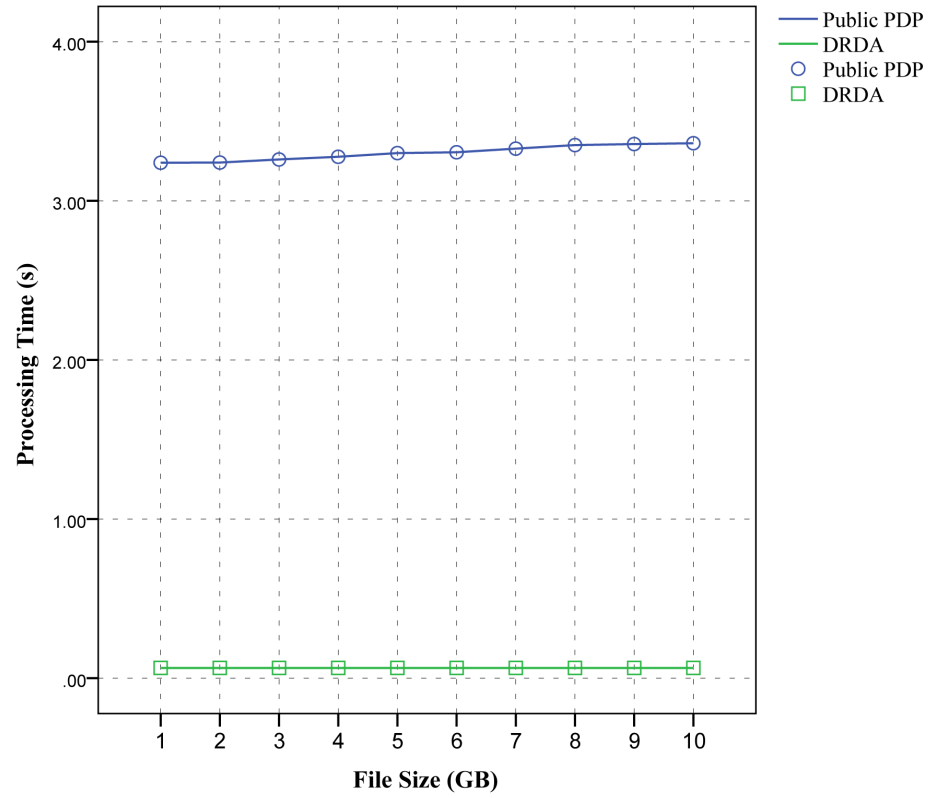


Figure 6.57: Comparison of processing time of Data Update when the number of updates is 4

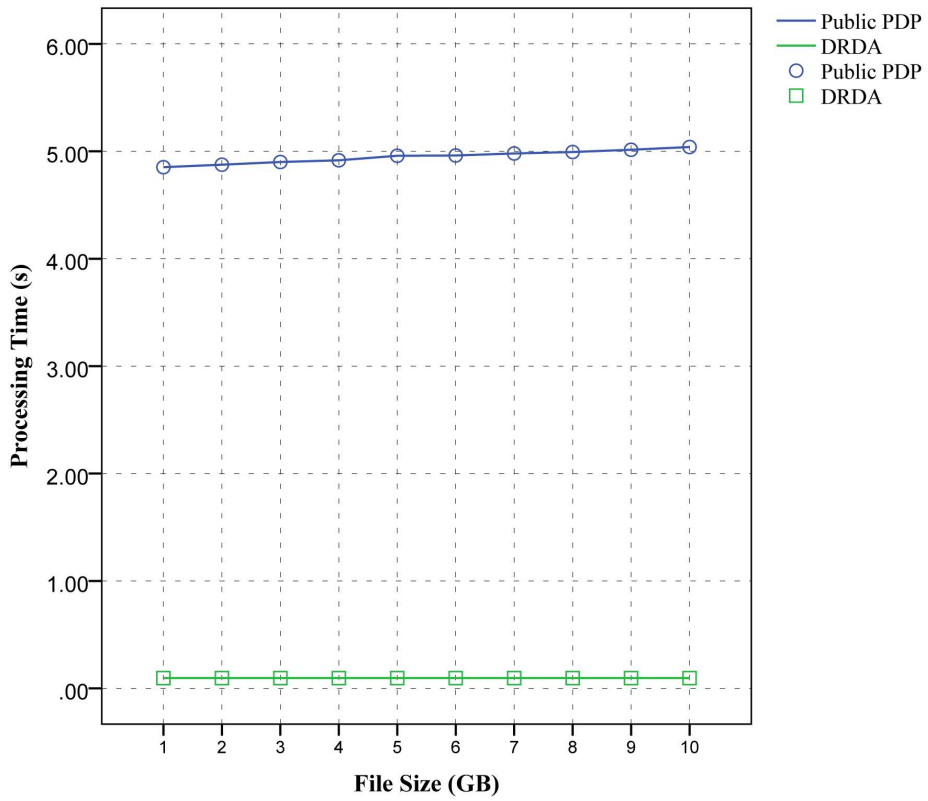


Figure 6.58: Comparison of processing time of Data Update when the number of updates is 6

processing time of the DRDA method is around 0.1 second.

The processing time of dynamic update operations in the DRDA method is compared by processing time of public PDP in Figure 6.59 when the number of update is 8. It can be seen that the proposed method significantly decreases the processing time of dynamic update for large-scale file.

Figure 6.60 shows the comparison of processing time of dynamic update in the DRDA and public PDP method when the number of update is 10. The experimental result indicates that the processing time of dynamic update in public PDP is around 8.26 second while the processing time of the DRDA method is about 0.18 second.

Figure 6.61 compares the processing time of dynamic data update operations in the DRDA and public PDP methods when the number of update is in the range of 2 to 10. The experimental result indicates that the processing time of the DRDA method is better than the processing time of public PDP method. It can be seen that by increasing the number

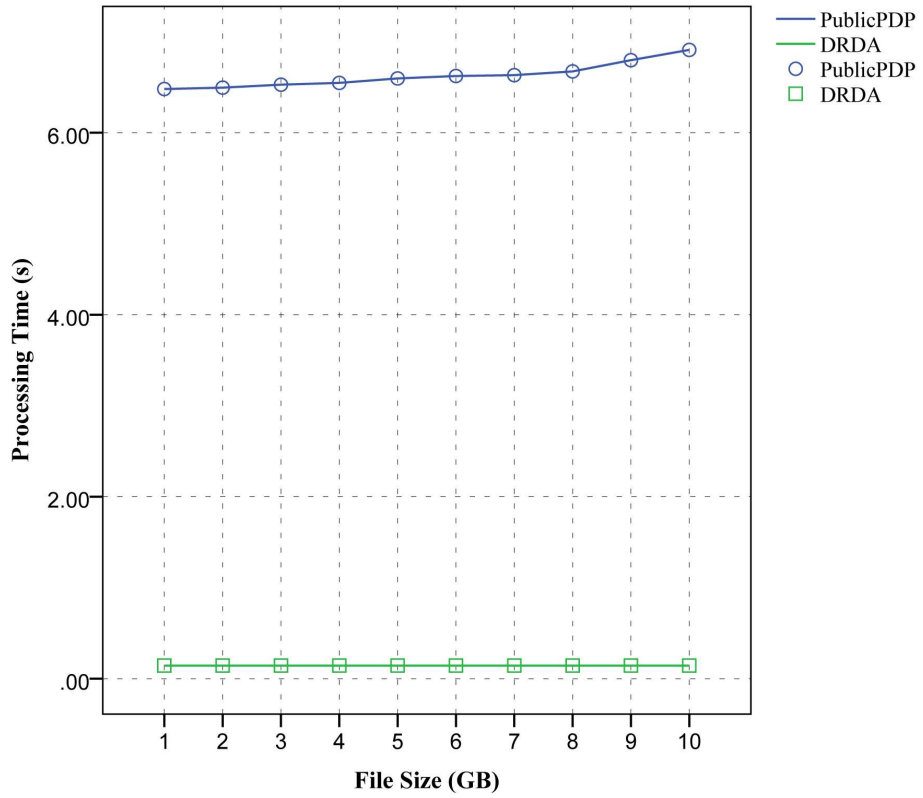


Figure 6.59: Comparison of processing time of Data Update when the number of updates is 8

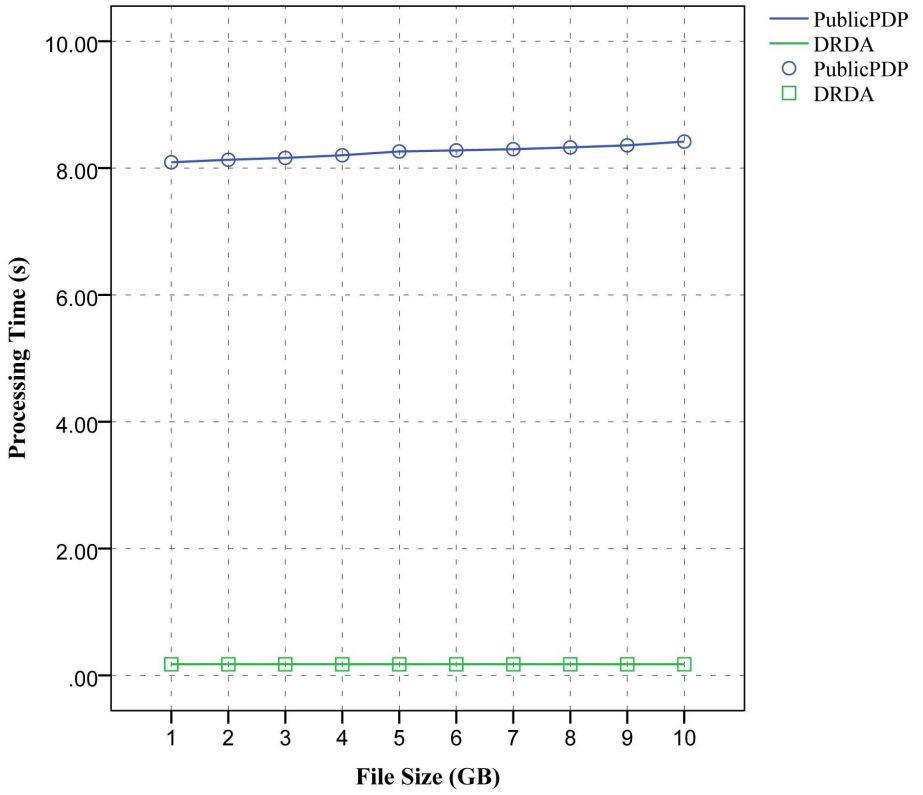


Figure 6.60: Comparison of processing time of Data Update when the number of updates is 10

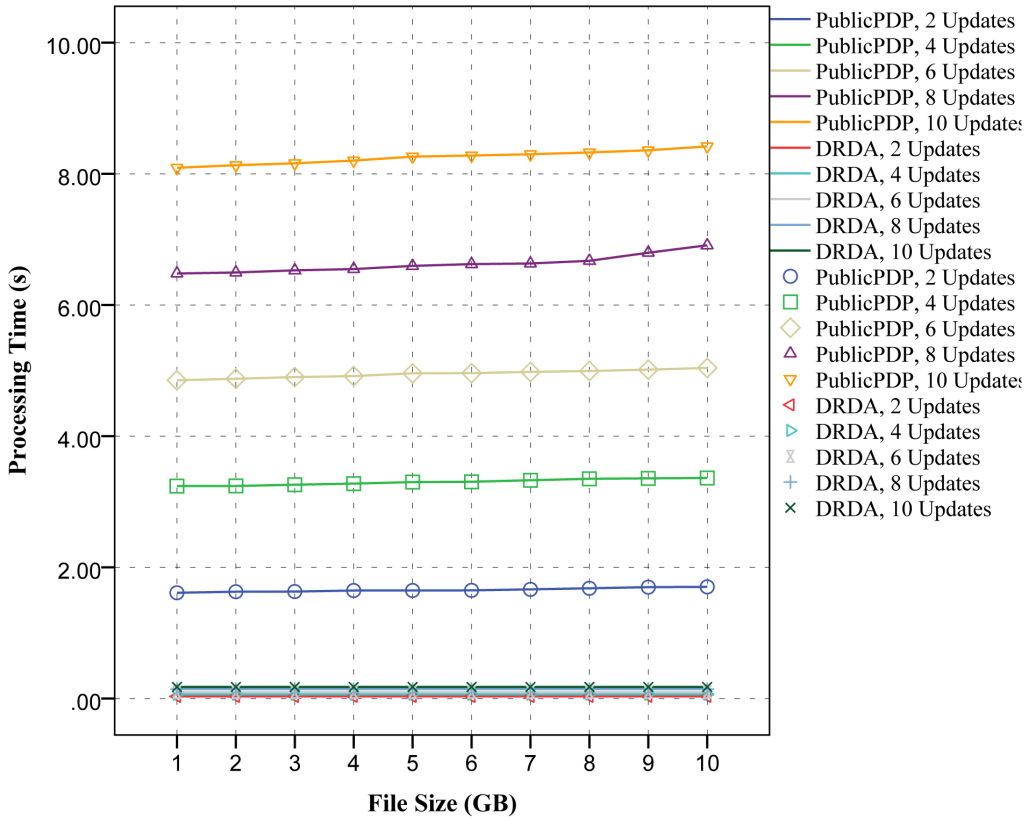


Figure 6.61: Comparison of processing time of Data Update for different number of updates

of update operation the processing time of public PDP method increases significantly. For example, the processing time of public PDP increases from 1.61 second to 8.26 second. The rate of processing time of update operations in the DRDA method is from 0.03 to 0.18 for different number of update operations.

6.7.4 Comparison of processing time of frequent data update

In this section, the effectiveness of the proposed data structure (D&CT) to support frequent dynamic data update operations is evaluated on the basis of the amount of the processing time of node re-balancing. To achieve this goal, different scenarios were designed and the result of each scenario was compared with the state-of-the-art methods to validate the proposed method.

In the first scenario, the experiments was conducted for 100 – 1000 times updating an outsourced file with length 1 GB, including 125,000 data blocks. The main reason to

perform such an experience is to demonstrate the effectiveness of the proposed method for frequent data update. The update operation consists of inserting or deleting a data block randomly (Ateniese et al., 2011; L. Chen, Zhou, Huang, & Xu, 2013; Erway et al., 2009; Q. A. Wang et al., 2011).

The first method that contributes in this scenario is Public PDP (Q. A. Wang et al., 2011; C. Wang, Wang, et al., 2012)(Wang et al., 2011) that used the MHT for supporting dynamic data update. As explained in section 2.4.1.2, the Wang scheme is one of the best and well-known data auditing methods in terms of dynamic data update operation. To insert or delete a block in such scheme, the auditor needs to find the position of the block (i) in the MHT tree. Moreover, inserting or deleting a block requires re-calculating the hash of the new leaf and existing nodes in the path to the root of the tree each time that incurs the huge computation overhead on the auditor. The next method in this scenario is EPP-PDP method (Yang & Jia, 2013), which supports dynamic data update using an index table to support dynamic data update operations. Similarly, after finding the position of the block (i) in the EPP-PDP method, the auditor prerequisites to shift the remaining $(n-i)$ blocks for every insert or delete operation. Subsequently, repeating this process multiple times results in a significant computation overhead on the auditor.

As aforementioned in section 4.3 of chapter 4, the optimal number of division in the proposed method is equal to $\sqrt{n}$, where n is the number of blocks. As a result, the number of division must be 353 approximately. Therefore, the experiment of the proposed scheme is performed when 353 D&CTs with size 355 are used for dynamic updates. By using the D&CT data structure, the number of shifts reduced in DRDA method resulting in the minimum computation overhead on the client side.

Figure 6.62 shows the performance in terms of processing time under the different number of update (insert or delete) operations. it can be understood that the D&CT is able to reduce the processing time of data update.

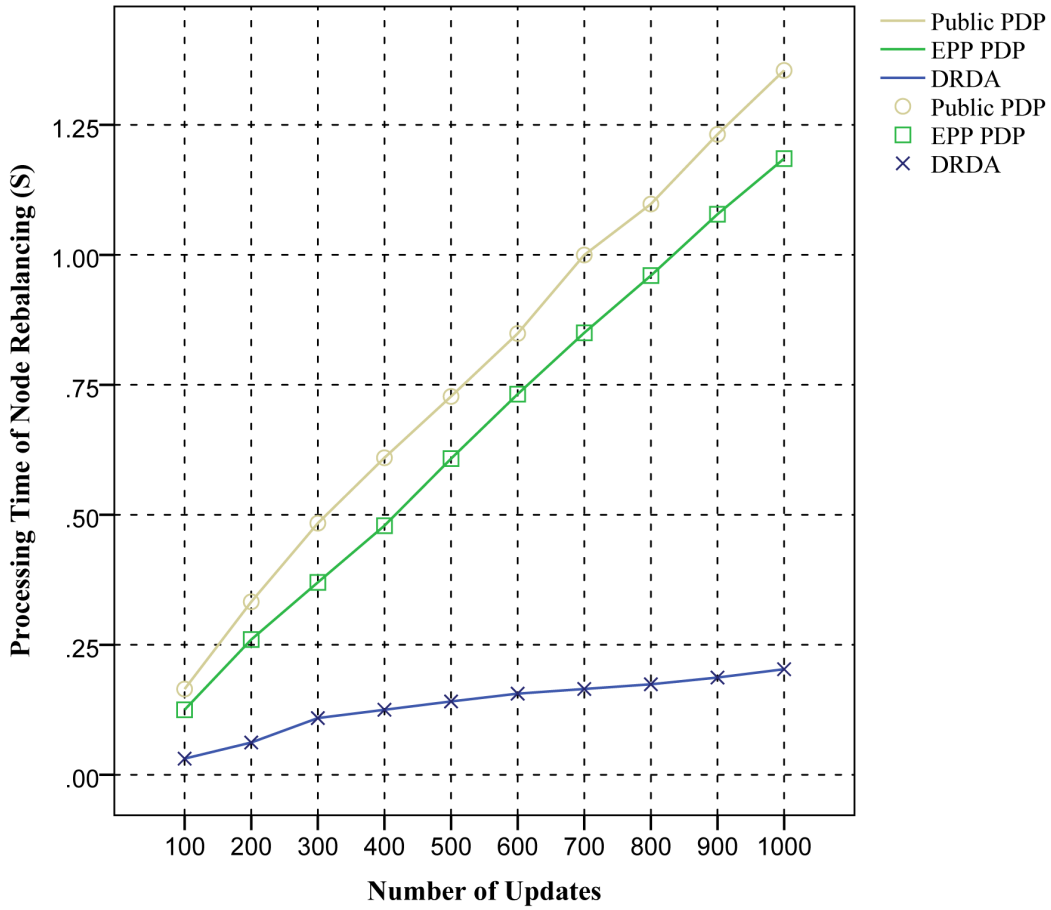


Figure 6.62: The Comparison of Processing Time of Node Re-balancing during Different Number of Update Operations

The ANOVA test was also used to ensure the validity of the data and comparisons. Firstly, the normality of data is checked on the basis of Shapiro-Wilk test $(P < 0.05)$. The results confirm that all data were normal (Significant of public PDP method: 0.936, significant of EPP PDP method: 0.879, and significant of the proposed method: 0.578394). The ANOVA analysis also approved a significant difference between the proposed scheme and the other methods. Table 6.3 shows the validity of the comparison of the processing time for different number of update operations using Post Hoc tests.

The second scenario demonstrated the effect of the data update on the different large-scale files (with sizes from 1 GB to 10 GB) in which the data owner updates the outsourced file 10 times and 100 times (Figure 6.63).

Figure 6.63-a shows when the number of updates are 10, the computation overhead

Table 6.3: Validity of the Comparison of Processing Time for Different Number of Update Operations using Post Hoc tests

Multiple Comparisons

Dependent Variable: Processing Time

Games-Howell

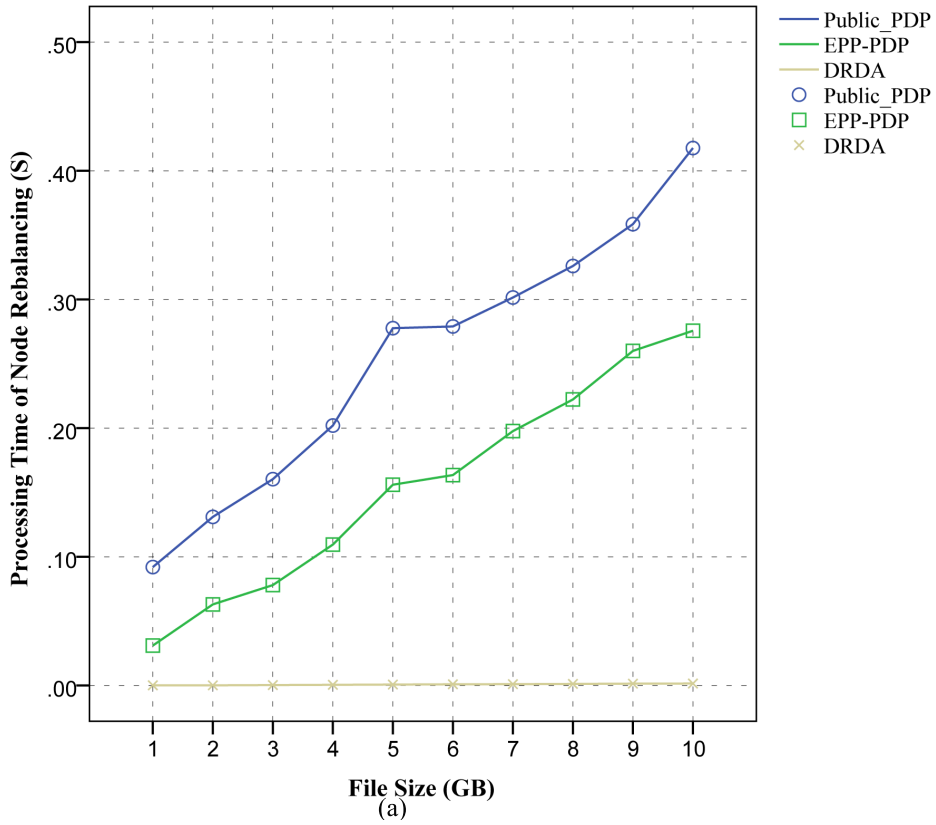
(I) Methods	(J) Methods	Mean Difference (I-J)	Sig.
Public PDP	EPP PDP	0.12036	0.757
	DRDA	.6452625*	0.001
EPP PDP	Public PDP	-0.1204	0.757
	DRDA	.5249000*	0.003
DRDA	Public PDP	-.6452625*	0.001
	EPP PDP	-.5249000*	0.003

*. The mean difference is significant at the 0.05 level.

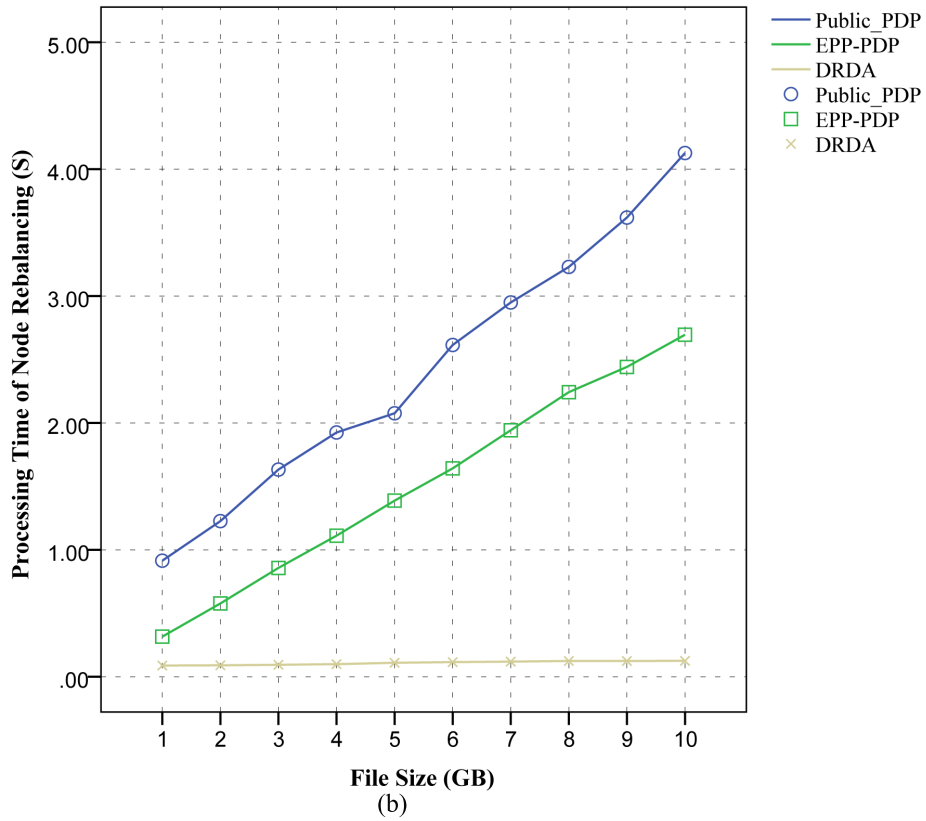
of the public PDP method (Wang et al., 2011) is increasing from 0.0920 second to approximately 0.4177 second by increasing the size of file. By increasing the number of updates to 100, the processing time of node re-balancing in the public PDP method raised more (from 0.9143 to 4.1273). This is because the auditor encounters a huge number of data block in the MHT. Similarly, in the EPP-PDP method, when the size of input file is enhancing from 1 GB to 10 GB with the same size of data block (8 kB), the number of data blocks are also increasing. Consequently, the auditor requires shifting huge numbers of blocks to insert or delete a data block. For example, the processing time for updating 10 blocks is from 0.0310 second to 0.2757 second.

As it is shown in Figure 6.63-b, the DRDA method incurs the minimum overhead on the auditor (maximum 0. 0015 second when the number of updates are 10, and 0.1250 second for performing update operations 100 times). The main reason for decreasing the processing time of node re-balancing in the DRDA method is using the structure of D&CTs in the proposed data auditing method (number of divisions = 353). Therefore, the DRDA method can be applicable for auditing large scale files dynamically.

To ensure the normality of data for each method as a preliminary assumption for ANOVA, the Shapiro-Wilk test $(P < 0.05)$ carried out and the result confirms the normality of data. The conducted ANOVA analysis shows that the proposed scheme has a



(a)



(b)

Figure 6.63: The Effect of Large-Scale Files on Processing Time of Node Re-balancing during Dynamic Update Operation When (a) Number of Updates = 10, (b) Number of Updates = 100

significant difference $(p = 0.05, Sig = 0.01)$ with EPP PDP and public PDP schemes. Table 6.4 shows the validity of the comparison using Post Hoc tests.

Table 6.4: Validity of the Comparison of Processing Time for Large Scale files using Post Hoc tests

Multiple Comparisons
Dependent Variable: Processing Time
Games-Howell

(I) Methods	(J) Methods	Mean Difference (I-J)	Sig.
Public PDP	EPP PDP	0.0948	0.819
	DRDA	.65125*	0.001
EPP PDP	Public PDP	-0.0948	0.819
	DRDA	.55645*	0.001
DRDA	Public PDP	-.65125*	0.001
	EPP PDP	-.55645*	0.001

*. The mean difference is significant at the 0.05 level.

6.7.5 Comparison of Complexity of Communication Cost and Processing Time

Table 6.5 shows a comparison of our scheme and state-of-the-art remote data auditing protocols based on the communication and the computation overhead through dynamic data update, where *m* is the number of blocks, *n* is the number of sectors of a block; *c* indicates the number of challenge blocks in each auditing query, and *k* indicates the number of the D&CTs (Sookhak, Akhunzada, Gani, Khurram Khan, & Anuar, 2014). In this table, the processing time consists of: computation audit, computation modify, computation insert, computation delete, and computation append. As mentioned earlier in section 5.2.3 of chapter 5, the processing time of the response phase is incurred on the server, and the processing time of the verification phase is incurred on the auditor. Moreover, the incurred processing time on the auditor during modify, insert, delete, and append operation are called computation modify, computation insert, computation delete, and computation append respectively.

From the table, it can be found that the public PDP method (Q. A. Wang et al., 2011; C. Wang, Wang, et al., 2012)) has the maximum computation overhead during dynamic

data update. It is because this method uses the MHT data structure to check the integrity or perform update operations on the outsource data blocks.

Although modifying and appending a block in the EPP PDP scheme (Yang & Jia, 2013) is efficient $(O(c))$, to insert a block after i or delete a specific block $(f[i])$, the verifier must shifts $(m-i)$ entities in the data structure. As a result, the computation overhead of such method during insert and delete operations is $O(m)$ (Sookhak, Akhunzada, et al., 2014).

This study carried out to address this issue and improve the auditing scheme by designing a new data structure (D&CT) for reducing the computation overhead. As mentioned earlier in section 4.2, the verifier only needs to shift $(\frac{m}{k}-i)$ blocks that incurs $O(\frac{m}{k})$ computation overhead on the verifier. It is important to mention that to find a block $(f[i])$ in D&CT structure, the verifier only needs to divide the location of block to k and find the appropriate D&CT that incurs negligible overhead on the verifier.

The first step to perform insert, delete, append, and modify operations is to identify that the i^{th} block of data block is stored in which D&CTs. The auditor is able to find the i^{th} data block by computing the quotient of a division of the requested block index (i) by the number of data block in each D&CT structure (k). Such quotient shows the D&CT number and the remaining of the division shows the position of block in the found D&CT. To insert a new data block after j^{th}data block or delete the j^{th} data block, the auditor has to find the considered D&CT and the position of the block in it (i), and then moves forward or backward the remain blocks of the D&CT $(\frac{m}{k}-i)$.Since each D&CT contains $(\frac{m}{k})$ blocks, performing insert and delete operations incur $O(\frac{m}{k})$ computation overhead on the auditor. The modification operation incurs $O(C)$ as a computation overhead on the auditor. It is because the auditor only requires finding the position of i^{th} data block in the D&CTs and modifying the content. Finally, to perform append operation, the auditor must inset a new data block after the last data block of the last D&CT which

Table 6.5: The comparison of different remote data auditing schemes

Methods	Comm. Cost	Processing Time					
		Verify		Modify	Insert	Delete	Append
		Server	Client				
Public PDP	$O(c\log m)$	$O(c\log m)$	$O(c\log m)$	$O(c\log m)$	$O(c\log m)$	$O(c\log m)$	$O(c\log m)$
EPP PDP	$O(c)$	$O(cn)$	$O(c)$	$O(c)$	$O(m)$	$O(m)$	$O(c)$
DRDA	$O(c)$	$O(cn)$	$O(c)$	$O(c)$	$O(\frac{m}{k})$	$O(\frac{m}{k})$	$O(c)$

impose $O(C)$ as a processing time. Furthermore, for auditing the outsourced blocks, the auditor needs to send a challenge message including indices of some blocks or receiving a response message including integration of requested blocks and tags, which incur constant communication cost on the auditor.

6.8 Conclusion

This chapter presents the experimental result of the proposed remote data auditing method to prove the efficiency of the DRDA method on the basis of the processing time and communication cost.

The experiment was conducted based on the significant criteria, such as: size of file, size of signature, and probability of detection. The processing time tested during four different phases of the proposed scheme: setup, challenge, response, and verification phases. However, since the data owner only requires to perform the setup phase one time, the computation overhead for setup phase was negligible. The communication cost of the method is also examined during three phases: setup, challenge, and response phases. The main goals of the implementation are: (1) to identify the effect of size of file, size of signature, and probability of detection on the computation and communication overhead of the proposed scheme, (2) to find the effective criteria for the proposed scheme that incur minimum overhead on the auditor.

Finally, to validate the implementation of the results, we compared the DRDA scheme

with the state-of-the-art methods on the basis of the processing time and the communication cost. We also validated the results by using the one-way analysis of variance (ANOVA) analysis to study whether there are any significant differences between the means of experimented methods or not. The next chapter of this study concludes the research, makes an overview of the objectives and how to fulfill them, and presents the scope, limitation, and some open issues for further investigations.

CHAPTER 7

CONCLUSION

7.1 Introduction

This chapter concludes the research on proposing a dynamic remote data auditing scheme for securing big data storage in cloud computing. For this reason, the research objectives are re-evaluated and the accomplishments of this study are put forward. Moreover, the contributions of the research are summarized and the limitations and scope of this work are discussed. Finally, some future research directions of this research are presented.

The complete organization of this chapter is as follows. Section 7.2 provides an overview of the results and finding of this study to show how the research objectives are fulfilled. The contributions of this research are highlighted in section 7.3. In section 7.4, the scope and limitations are discussed. Section 7.5 presents some outstanding issues as a future work.

7.2 Research Summary and Objectives Achievement

The main purpose of this study was to investigate the problem of additional processing time in the existing data auditing methods and develop a remote data auditing method that can be used to check the integrity of the outsourced data in cloud computing. This scheme has the capability to securely support dynamic data update operations on the block level, such as insert, delete, modify, and append operations. As mentioned earlier in section 1.4, this research has four objectives that are achieved as follows:

To identify the gaps and outstanding issues in the area of data storage integrity of cloud computing, the literature review was performed as a first objective of this research.

We selected more than 300 papers from the most important web-based databases and online digital libraries such as IEEE, ACM, Elsevier, Springer, Wiley, and ISI Web of Science. These papers were within the domain of cloud computing, mobile cloud computing, security and privacy in cloud and mobile cloud computing, modern cryptographic protocols (including Homomorphic mechanism, algebraic signature) with in 7 years (2007 – 2014). We proposed a thematic taxonomy on the basis of the state-of-the-art data auditing methods to meet the requirements of the literature review objective. The remote data auditing schemes are classified into three main groups in the thematic taxonomy (section 2.3.2). We also used qualitative analysis to compare the existing methods and highlight the advantages and disadvantages of them (section 2.5). Finally, the open issues and challenges of data auditing schemes in cloud and mobile cloud computing environment that have not been addressed yet were identified and highlighted (section 2.6).

The quantitative analysis based on computation overhead and communication overhead as two important measures for auditing methods was utilized to establish and justify the research problem. The existing data auditing approaches were implemented in the real cloud computing environment, and the benchmark test was used to evaluate such methods based on the computation and communication cost on the client and server side (section 3.2). Moreover, the impact of dynamic data update operations was analyzed on the existing data approaches in the real environment (section 3.3). We also studied the effect of dynamic data update operations on the large-scale file size (section 3.4). Finally, the impact of frequent data updates was evaluated for different size of the files (section 3.5).

A new remote data auditing method was proposed on the basis of algebraic signature technique to fulfill the objective of efficient solution for checking the integrity of the outsourced data in cloud computing. The proposed scheme addresses the problem of additional computation and communication cost for cloud data storage system. We also

designed a new data structure to support dynamic data update with minimum computation and communication cost on the auditor. In other words, by using the D&CT data structure, the data owner has the capability to modify, delete, insert, or append in block level without requiring to download the whole file. The D&CT data structure also empowers our method to be applicable for large-scale data with least processing time on the client.

The proposed data auditing scheme is implemented in the real environment by using java and C++ language to address the objective of evaluating DRDA method. The performance of the DRDA scheme was validated by using the benchmark test in the emulation environment.

We analyzed the DRDA scheme by using distinctive parameters such as length of signature, file size, and probability of detection. The different scenarios also defined to evaluate the proposed method. Furthermore, we analyzed the strength of the security on the basis of mathematic to validate and proof the security of the DRDA method (section 6.2 and section 6.3). The comparison of the results with the sate-of-the-art remote data auditing method validated the efficiency of the proposed method in terms of computation and communication cost (section 6.6). The results showed that the D&CT data structure reduces the processing time of dynamic data update operations by decreasing the number of shifting. The results also demonstrated that the D&CT data structure dramatically decrease the processing time of dynamic data update for large-scale outsourced file in cloud computing.

7.3 Contribution of the Research

After focusing on the existing data auditing methods, the problems were identified: (1) additional computation overhead through checking the integrity of outsourced data, (2) additional communication and computation overhead through dynamic data update operations for normal file size and large-scale file size, and (3) additional computation

overhead through frequent data update (chapter 3). Accordingly, the contributions of this research are summarized as follows:

Thematic Taxonomy: The first contribution of this study is to design a thematic taxonomy for data auditing schemes in cloud computing environment. We analyzed and classified the current RDA approaches into three different categories on the basis of the data redundancy feature, namely: integrity-based, recovery-based, and deduplication-based approaches. We aimed to investigate the similarities and differences of such schemes based on the thematic taxonomy to diagnose the significant and outstanding issues for further investigation.

Effective Static Remote Data Auditing (SRDA) Method: The next contribution of this research is to propose an effective remote data auditing method for preserving the integrity of static data in the cloud computing. The main idea behind of this scheme is to utilize the algebraic properties of the outsourced data blocks to remotely check the integrity of files. The experimental results showed that the proposed scheme incurs minimum computation and communication cost on the auditor. Since the size of outsourced file has negligible computation overhead on the auditor, the SDRA method has the capability to be applied for different size of file.

Divide and Conquer Table (D&CT): Although the SRDA method is able to check the integrity of the outsourced data in the cloud storage system, modifying a single block can incur significant processing time on the auditor. For example, to insert a new block after $b[i]$, the data owner must download $n-i$ block, modify the tag of them, insert the new block and upload the $n+1-i$ block to the cloud storage. It is clear that performing such process imposes high computation and communication overhead on the client and server. In addition, such modification allows the attacker to perform the replay attack by passing the verification phase using an old version

of the file. We design a new data structure, namely D&CT to address this problem and support dynamic data update operation efficiently.

Efficient Dynamic Remote Data Auditing (DRDA) Method: To support dynamic data update, the SRDA scheme was modified on the basis of the D&CT in which the data owner has the capability to update the outsourced data in the block level. In other words, performing an update operation on a single block is independent of the other blocks to reduce the computation and communication overhead on the client and server side. As a result, the data owner only needs to download the request data block and upload it after accomplishing the update operation.

Updating large-Scale-file size: One of the main advantages of the DRDA method is to support dynamic data update for large-scale file size with minimum processing time on the data owner. For example, inserting a data block in the large-scale file (100 GB file with 26214400 blocks) requires shifting a large number of data blocks, which incur high processing time on the auditor. The D&CT is designed in such a way that a small part of data blocks, including the requested block must be shifted to reduce the computation overhead.

Frequently data update: The DRDA scheme allows the data owner to modify a small part of file frequently with minimum processing time. Therefore, this method is applicable for social network systems in which the users are able to frequently modify a part of file.

In the rest of this section, some research papers are outlined:

Accepted Articles:

- Mehdi Sookhak, Abdullah Gani, Samee U. Khan, Rajkumar Buyya, Albert Y. Zomaya, “Remote Data Auditing in Cloud Computing Environment,” ACM Com-

puting Surveys, Accepted (ISI Indexed Q1, Impact Factor 4.1, ERA Ranking A*)

- Mehdi Sookhak, Hamid Talebian, Ejaz Ahmed, Abdullah Gani, Muhammad Khurram Khan, “A review on remote data auditing in single cloud server: Taxonomy and open issues”, Journal of Network and Computer Applications, Volume 43, August 2014, Pages 121-141, ISSN 1084-8045, http://dx.doi.org/10.1016/j.jnca.2014.04.011 (ISI Indexed Q1, Impact Factor 1.772, ERA Ranking A). In the list of top most downloaded paper in journal of Network and Computer Applications.

- Mehdi Sookhak, Adnan Akhunzada, Abdullah Gani, Muhammad Khurram Khan, Nor Badrul Anuar “ Towards Dynamic Remote Data Auditing in Computational Clouds,” Scientific World Journal, Accepted 8 May 2014,(ISI Indexed Q1, Impact Factor 1.730).

- Mehdi Sookhak, Abdullah Gani, Muhammad Khurram Khan, “ Geographic Wormhole Detection in Wireless Sensor Network,” Plos One, Accepted (ISI Indexed Q1, Impact Factor 3.7, ERA Ranking A)

- Muhammad Shiraz, Mehdi Sookhak, Abdullah Gani, Syed Adeel Ali Shah, “A Study on the Critical Analysis of Computational Offloading Frameworks for Mobile Cloud Computing”, Journal of Network and Computer Applications, Volume 47, January 2015, Pages 47-60, ISSN 1084-8045, (ISI Indexed Q1, Impact Factor 1.772, ERA Ranking A).

- Adnan Akhunzada, Mehdi Sookhak, Nor Badrul Anuar, Abdullah Gani, Steven Furnell, Amir Hayat, Muhammad Khurram Khan, “Man-At-The-End attacks: Analysis, taxonomy, human aspects, motivation and future directions”, Journal of Network and Computer Applications, Volume 48, February 2015, Pages 44-57, ISSN 1084-8045, (ISI Indexed Q1, Impact Factor 1.772, ERA Ranking A).

- Md Whaiduzzaman, Mehdi Sookhak, Abdullah Gani, Rajkumar Buyya, “A survey on vehicular cloud computing”, Journal of Network and Computer Applications, Volume 40, April 2014, Pages 325-344, ISSN 1084-8045, (ISI Indexed Q1, Impact Factor 1.772, ERA Ranking A). Second place in the list of top most downloaded paper in journal of Network and Computer Applications with 54 citations.

- Ejaz Ahmed, Abdullah Gani, Mehdi Sookhak, Siti Hafizah, Feng Zia “Application Optimization in Mobile Cloud Computing: Motivation, Taxonomies, and open challenges”, Journal of Network and Computer Applications, In press (ISI Indexed Q1, Impact Factor 1.772, ERA Ranking A).

Under Review Articles:

- Mehdi Sookhak, Abdullah Gani, Muhammad Khurram Khan, Rajkumar Buyya, “Performance Analysis of Dynamic Remote Data Auditing for Cloud Computing,” Information Sciences Journal, Under Review (ISI Indexed Q1, Impact Factor 3.89, ERA Ranking A)

- Mehdi Sookhak, Abdullah Gani, Yang Xiang, Cong Wang, Muhammad Khurram Khan, Rajkumar Buyya, “ Dynamic Remote Data Auditing for Securing Data Storage in Cloud Computing,” IEEE Transactions on Services Computing, Under Review (ISI Indexed Q1, Impact Factor 1.985, ERA Ranking A*)

- Mehdi Sookhak, Abdullah Gani, Muhammad Khurram Khan, Rajkumar Buyya, “ Attribute-Based Data Access Control in Cloud Computing Taxonomy and Open Issues,” Pervasive and Mobile Computing Journal, Under Review (ISI Indexed Q2, Impact Factor 1.66, ERA Ranking A)

- Mohammad Reza Jabbarpour, Mehdi Sookhak, Rafidah Md Noor , Abdullah Gani, Chi Harold Liu, Kin K. Leung, “A Survey on Cloud-enabled Vehicular Networks:

Architectures, Applications and Open Issues", IEEE Communication Survey and Tutorials, Under Review, (ISI Indexed Q1, Impact Factor 6.49, ERA Ranking A*).

7.4 Research Scope and Limitations

The scope of this study is restricted to two main parts: (1) analyzing the problem of existing data storage integrity methods for the cloud storage, and (2) proposing a new remote data auditing method with minimum computation overhead. The limitation of this study also listed as follows:

- This research only focuses on single provable data possession (PDP) methods wherein a singular copy of the file is outsourced in the cloud storage. Therefore, the distributed auditing methods are not considered in this research.

- The proposed scheme can be used for checking the integrity of two types of file size such as normal size (10 MB – 50 MB) and large-scale size (1 GB – 10 GB), which are extracted from the existing literature. For example, the maximum file size for storing documents in Google Drive is 10 MB to 50 MB (*Google Docs, Sheets, and Slides size limits*, 2015), Box is 2 GB to 5 GB (*What's the maximum file size I can upload?*, 2014), and OneDrive is 2 GB to 10 GB (Perez, 2014).

7.5 Future Works

The RDA methods are applicable for the single and distributed servers (B. Chen et al., 2010). In the single server that only a copy of the file is outsourced, such algorithms are only responsible to prevent unauthorized parties from altering the outsourced data. In other words, the auditor must check the data integrity through a RDA algorithm to detect data corruption (Ateniese et al., 2011; B. Chen & Curtmola, 2012; Cash et al., 2012; Erway et al., 2009). However, when data corruption is detected, the majority of the single

server RDA techniques do not have the necessary capabilities to recover data (Sookhak et al., 2015).

Currently, individuals and organizations prefer to store data on distributed servers, because the single server setting does not support data recovery when data corruption is detected. For instance, in deep archival applications that use peer-to-peer storage systems (Maniatis, Roussopoulos, Giuli, Rosenthal, & Baker, 2005), the third party is responsible for managing the data. Therefore, data owners need RDA to verify the integrity and correctness of the large archival data sets, which makes the single server auditing methods prohibitive. This is because most of the aforementioned RDA approaches are inapplicable to such systems, or incur huge computation and communication overhead on the client and server (Sookhak et al., 2015). Consequently, the RDA technique is complemented with data storage redundancy on multiple servers because the data owner is able to restore the corrupted data by using the remaining healthy servers (Sookhak et al., 2015; B. Chen et al., 2010).

This study is focused on proposing a single remote data auditing method with minimum processing time and communication overhead. However, the proposed scheme lacks in considering distributed systems issues. Hence, the future work includes improving the DRDA method to be applicable for auditing the integrity of large archival files in the distributed storage system.

Appendices

APPENDIX A

LIST OF PUBLICATIONS

Accepted Articles:

- ***Mehdi Sookhak***, Abdullah Gani, Samee U. Khan, Rajkumar Buyya, Albert Y. Zomaya, "Remote Data Auditing in Cloud Computing Environment," ACM Computing Surveys, Accepted (ISI Indexed Q1, Impact Factor 4.043, 5-years Impact Factor 7.443, ERA Ranking A*). ACM Computing Surveys is the best ISI journal in Computer Science, Theory & Method category.

- ***Mehdi Sookhak***, Hamid Talebian, Ejaz Ahmed, Abdullah Gani, Muhammad Khurram Khan, "A review on remote data auditing in single cloud server: Taxonomy and open issues", Journal of Network and Computer Applications, Volume 43, August 2014, Pages 121-141, ISSN 1084-8045, (ISI Indexed Q1, Impact Factor 1.772, ERA Ranking A). In the list of top most downloaded paper in journal of Network and Computer Applications.

- ***Mehdi Sookhak***, Adnan Akhunzada, Abdullah Gani, Muhammad Khurram Khan, Nor Badrul Anuar " Towards Dynamic Remote Data Auditing in Computational Clouds," Scientific World Journal, Accepted 8 May 2014,(ISI Indexed Q1, Impact Factor 1.730).

- ***Mehdi Sookhak***, Abdullah Gani, Muhammad Khurram Khan, " Geographic Wormhole Detection in Wireless Sensor Network," Plos One, Accepted (ISI Indexed Q1, Impact Factor 3.7, ERA Ranking A)

Under Review Articles:

- ***Mehdi Sookhak***, Abdullah Gani, Yang Xiang, Cong Wang, Muhammad Khurram Khan, Rajkumar Buyya, " Dynamic Remote Data Auditing for Securing Data Storage in Cloud Computing," IEEE Transactions on Services Computing, Under Review (ISI Indexed Q1, Impact Factor 1.985, ERA Ranking A*)

- ***Mehdi Sookhak***, Abdullah Gani, Muhammad Khurram Khan, Rajkumar Buyya, "Performance Analysis of Dynamic Remote Data Auditing for Cloud Computing," Information Sciences Journal, Under Review (ISI Indexed Q1, Impact Factor 3.89, ERA Ranking A)

- ***Mehdi Sookhak***, Abdullah Gani, Muhammad Khurram Khan, Rajkumar Buyya, " Attribute-Based Data Access Control in Cloud Computing Taxonomy and Open Issues," Pervasive and Mobile Computing Journal, Under Review (ISI Indexed Q2, Impact Factor 1.66, ERA Ranking A)

Collaborative Articles:

- Muhammad Shiraz, ***Mehdi Sookhak***, Abdullah Gani, Syed Adeel Ali Shah, "A Study on the Critical Analysis of Computational Offloading Frameworks for Mobile Cloud Computing", Journal of Network and Computer Applications, Volume 47, January 2015, Pages 47-60, ISSN 1084-8045, (ISI Indexed Q1, Impact Factor 1.772, ERA Ranking A).

- Adnan Akhunzada, ***Mehdi Sookhak***, Nor Badrul Anuar, Abdullah Gani, Steven Furnell, Amir Hayat, Muhammad Khurram Khan, "Man-At-The-End attacks: Analysis, taxonomy, human aspects, motivation and future directions", Journal of Network and Computer Applications, Volume 48, February 2015, Pages 44-57, ISSN 1084-8045, (ISI Indexed Q1, Impact Factor 1.772, ERA Ranking A).

- Md Whaiduzzaman, ***Mehdi Sookhak***, Abdullah Gani, Rajkumar Buyya, “A survey on vehicular cloud computing”, Journal of Network and Computer Applications, Volume 40, April 2014, Pages 325-344, ISSN 1084-8045, (ISI Indexed Q1, Impact Factor 1.772, ERA Ranking A). Second place in the list of top most downloaded paper in journal of Network and Computer Applications with 50 citations.

- Ejaz Ahmed, Abdullah Gani, ***Mehdi Sookhak***, Siti Hafizah, Feng Zia “Application Optimization in Mobile Cloud Computing: Motivation, Taxonomies, and open challenges”, Journal of Network and Computer Applications, In press (ISI Indexed Q1, Impact Factor 1.772, ERA Ranking A).

- Mohammad Reza Jabbarpour, ***Mehdi Sookhak***, Rafidah Md Noor , Abdullah Gani, Chi Harold Liu, Kin K. Leung, “A Survey on Cloud-enabled Vehicular Networks: Architectures, Applications and Open Issues”, IEEE Communication Survey and Tutorials, Under Review, (ISI Indexed Q1, Impact Factor 6.49, ERA Ranking A*).

- Abdullah Gani, Golam Mokatder Nayeem, Muhammad Shiraz, ***Mehdi Sookhak***, Md Whaiduzzaman, Suleman Khan, A review on interworking and mobility techniques for seamless connectivity in mobile cloud computing, Journal of Network and Computer Applications, Volume 43, August 2014, Pages 84-102, ISSN 1084-8045, (ISI Indexed Q1, Impact Factor 1.772, ERA Ranking A).

- Abdullah Yousafzai, ***Mehdi Sookhak***, Abdullah Gani, Muhammad Khurram Khan, “ Cloud Resource Allocation Techniques: Review, Taxonomy, and Open Issues,” Knowledge and Information Systems Journal, Revision (ISI Indexed Q1, Impact Factor 2.67, ERA Ranking A)

- Mohammadreza Eslaminejad, Abd Razak Shukor, ***Mehdi Sookhak***,“ Classification of energy-efficient routing protocols for wireless sensor networks,” Ad hoc Sensor

Wireless Network, Volume 17, August 2013, Pages 103-129, ISSN 1551-9899, (ISI Indexed Q4, Impact Factor 0.478)

REFERENCES

Aceto, G., Botta, A., de Donato, W., & Pescapè, A. (2013). Cloud monitoring: A survey. *Computer Networks*, *57*(9), 2093–2115. Retrieved from http://www.sciencedirect.com/science/article/pii/S1389128613001084 doi: http://dx.doi.org/10.1016/j.comnet.2013.04.001

Adelsbach, A., Katzenbeisser, S., & Sadeghi, A.-R. (2002). Cryptography meets watermarking: detecting watermarks with minimal or zero knowledge disclosure (EUSIPCO'02). In *Xi european signal processing conference* (Vol. 1, pp. 446–449). Toulouse, France.

AdelsonVelskii, M., & Landis, E. M. (1963). *An algorithm for the organization of information.* DTIC Document.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., ... Stoica, I. (2010). Above the Clouds: A View of Cloud Computing. *Communications of the ACM*, *53*(4), 50–58.

Arrington, M. (2006). *Gmail disaster: Reports of mass email deletions* (No. 25th Sep 2013). http://techcrunch.com/2006/12/28/gmail- disaster- reports-of- mass- email-deletions/.

Ateniese, G., Burns, R., Curtmola, R., Herring, J., Khan, O., Kissner, L., ... Song, D. (2011). Remote data checking using provable data possession. *ACM Trans. Inf. Syst. Secur.*, *14*(1), 1–34. doi: 10.1145/1952982.1952994

Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., & Song, D. (2007). Provable data possession at untrusted stores. In *Proceedings of the 14th acm conference on computer and communications security* (pp. 598–609). Alexandria, Virginia, USA: ACM. doi: 10.1145/1315245.1315318

Ateniese, G., Pietro, R. D., Mancini, L. V., & Tsudik, G. (2008). *Scalable and efficient provable data possession.* Istanbul, Turkey: ACM. doi: 10.1145/1460877.1460889

Baker, L. B., & Finkle, J. (2011). Sony PlayStation suffers massive data breach. *Reuters, April*, *26*.

Baker, W., Hutton, A., Hylender, C. D., Pamula, J., Porter, C., & Spitler, M. (2011). 2011 data breach investigations report. *Verizon RISK Team, Available: www. verizonbusiness. com/resources/reports/rp_databreach-investigations-report-2011_en_xg. pdf*, 1–72.

Bao, F., Deng, R., & Zhu, H. (2003). Variations of Diffie-Hellman Problem. In S. Qing, D. Gollmann, & J. Zhou (Eds.), *Information and communications security* (Vol. 2836, pp. 301–312). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-540-39927-8_28 doi: 10.1007/978-3-540-39927-8_28

Barga, R., Gannon, D., & Reed, D. (2011). The Client and the Cloud: Democratizing

Research Computing. *IEEE Internet Computing*, *15*(1), 72–75. doi: 10.1109/MIC.2011.20

Bayer, R. (1972). Symmetric binary B-Trees: Data structure and maintenance algorithms. *Acta Informatica*, *1*(4), 290–306. Retrieved from http://dx.doi.org/10.1007/BF00289509 doi: 10.1007/BF00289509

Bellare, M., Guérin, R., & Rogaway, P. (1995). *XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions* (Vol. 963). Santa Barbara, California, USA: Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/3-540-44750-4_2 doi: 10.1007/3-540-44750-4_2

Blaze, M., & Strauss, M. (1998). Atomic proxy cryptography. In *Proc. eurocrypt'97*. Citeseer.

Boneh, D., Gentry, C., Lynn, B., & Shacham, H. (2003). Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *Advances in cryptology (eurocrypt)* (Vol. 2656, pp. 416–432). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/3-540-39200-9_26 doi: 10.1007/3-540-39200-9_26

Boneh, D., Lynn, B., & Shacham, H. (2004). Short Signatures from the Weil Pairing. *Journal of Cryptology*, *17*(4), 297–319. Retrieved from http://dx.doi.org/10.1007/s00145-004-0314-9 doi: 10.1007/s00145-004-0314-9

Bowers, K. D., Juels, A., & Oprea, A. (2009). *Proofs of retrievability: theory and implementation.* Chicago, Illinois, USA: ACM. doi: 10.1145/1655008.1655015

Broder, A. Z. (1993). Some applications of Rabin's fingerprinting method. *Sequences II: Methods in Communications, Security, and Computer Science*, 143–152. doi: 10.1.1.53.6172

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, *25*(6), 599–616. doi: http://dx.doi.org/10.1016/j.future.2008.12.001

Cachin, C. (2011). *Integrity and Consistency for Untrusted Services* (Vol. 6543). Nový Smokovec, Slovakia: Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-18381-2_1 doi: 10.1007/978-3-642-18381-2_1

Calheiros, R. N., Vecchiola, C., Karunamoorthy, D., & Buyya, R. (2012). The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid Clouds. *Future Generation computer systems*, *28*(6), 861–870. Retrieved from http://www.sciencedirect.com/science/article/pii/S0167739X11001397 doi: http://dx.doi.org/10.1016/j.future.2011.07.005

Canetti, R., Goldreich, O., & Halevi, S. (2004). The random oracle methodology, revisited. *J. ACM*, *51*(4), 557–594. Retrieved from http://doi.acm.org/10.1145/1008731.1008734 doi: 10.1145/1008731.1008734

Carter, J. L., & Wegman, M. N. (1979). Universal classes of hash functions. *Journal*

of Computer and System Sciences, *18*(2), 143–154. Retrieved from http://www.sciencedirect.com/science/article/pii/0022000079900448 doi: http://dx.doi.org/10.1016/0022-0000(79)90044-8

Cash, D., Küpçü, A., & Wichs, D. (2012). Dynamic Proofs of Retrievability via Oblivious RAM. *IACR Cryptology ePrint Archive*, 550. Retrieved from http://eprint.iacr.org/

Castro, M., & Liskov, B. (2002). Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, *20*(4), 398–461. doi: 10.1145/571637.571640

Cellan-Jones. (2009). *The Sidekick Cloud Disaster* (Vol. 2013) (No. 25th Sep 2013). http://www.bbc.co.uk/blogs/technology/2009/10/the_sidekick_cloud_disaster.html.

Chaum, D., & Evertse, J.-H. (1986). Cryptanalysis of des with a Reduced Number of Rounds. In H. Williams (Ed.), *in proceedings of advances in cryptology (crypto '85), lecture notes in computer science* (Vol. LNCS 218, pp. 192–211). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/3-540-39799-X_16 doi: 10.1007/3-540-39799-X_16

Chen, B., & Curtmola, R. (2012). Robust dynamic provable data possession. In *32nd international conference ieee on distributed computing systems workshops (icdcsw), 2012* (pp. 515–525). IEEE. doi: 10.1109/ICDCSW.2012.57

Chen, B., Curtmola, R., Ateniese, G., & Burns, R. (2010). *Remote data checking for network coding-based distributed storage systems.* Chicago, Illinois, USA: ACM. doi: 10.1145/1866835.1866842

Chen, H., & Lee, P. (2013). Enabling Data Integrity Protection in Regenerating-Coding-Based Cloud Storage: Theory and Implementation. *IEEE Transactions on Parallel and Distributed Systems*, *PP*(99), 1. doi: 10.1109/TPDS.2013.164

Chen, L., Zhou, S., Huang, X., & Xu, L. (2013). Data dynamics for remote data possession checking in cloud storage. *Computers & Electrical Engineering*, *39*(7), 2413–2424. Retrieved from http://www.sciencedirect.com/science/article/pii/S0045790613001845 doi: http://dx.doi.org/10.1016/j.compeleceng.2013.07.010

Chen, Y., & Sion, R. (2011). *To cloud or not to cloud?: Musings on costs and viability.* New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2038916.2038945 doi: 10.1145/2038916.2038945

Chow, R., Golle, P., Jakobsson, M., Shi, E., Staddon, J., Masuoka, R., & Molina, J. (2009). Controlling data in the cloud: outsourcing computation without outsourcing control. In *Proceedings of the 2009 acm workshop on cloud computing security* (pp. 85–90). ACM.

Christiansen, C. A., Kolodgy, C. J., Hudson, S., Pintal, G., & IDC. (2010). *Identity and Access Management for Approaching Clouds (White Paper)* (Tech. Rep.). MA, USA. Retrieved from http://www.ca.com/us/~/media/files/industryanalystreports/cloud_security_wp_236234.aspx

chun Wesley. (2011). *What is google app engine?* https://ep2012.europython.eu/conference/talks/google-app-engine-best-practices-latest-features.

Clark, G. C., & Cain, J. B. (1981). *Error-Correction Coding for Digital Communications*. New York: Perseus Publishing.

Cong, W., Kui, R., Wenjing, L., & Jin, L. (2010). Toward publicly auditable secure cloud data storage services. *IEEE Network*, *24*(4), 19–24. doi: 10.1109/MNET.2010.5510914

Daemen, J., & Rijmen, V. (2000). The Block Cipher Rijndael. In J.-J. Quisquater & B. Schneier (Eds.), *Proceedings of international conference on smart cards research and applications (cards'98), lecture notes in computer science (lncs)* (Vol. 1820, pp. 277–284). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/10721064_26 doi: 10.1007/10721064_26

Diesburg, S. M., & Wang, A.-I. A. (2010). A survey of confidential data storage and deletion methods. *ACM Comput. Surv.*, *43*(1), 1–37. doi: 10.1145/1824795.1824797

Dodis, Y., Vadhan, S., & Wichs, D. (2009). Proofs of Retrievability via Hardness Amplification. In O. Reingold (Ed.), *Theory of cryptography* (Vol. 5444, pp. 109–127). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-00457-5_8 doi: 10.1007/978-3-642-00457-5_8

Du, W., Murugesan, M., & Jia, J. (2010). *Uncheatable grid computing.* Hachioji, Tokyo, Japan: Chapman & Hall/CRC.

Dutta, R., Barua, R., & Sarkar, P. (2004). Pairing-based cryptography: A survey. *Cryptology Research Group, Stat-Math and Applied Statistics Unit*, *203*.

Ekdahl, P., & Johansson, T. (2003). A new version of the stream cipher SNOW. In *In proceedings of the 9th international workshop on selected areas of cryptography (sac'02), lecture notes in computer science (lncs)* (Vol. 2595, pp. 47–61). Springer.

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory,*, *31*(4), 469–472.

Erway, C., Küpçü, A., Papamanthou, C., & Tamassia, R. (2009). *Dynamic provable data possession.* Chicago, Illinois, USA: ACM. doi: 10.1145/1653662.1653688

Esiner, E., Kachkeev, A., & Ozkasap, O. (2013). *FlexList: Optimized Skip List for Secure Cloud Storage* (Tech. Rep.).

Fernando, N., Loke, S. W., & Rahayu, W. (2013). Mobile cloud computing: A survey. *Future Generation computer systems*, *29*(1), 84–106. Retrieved from http://www.sciencedirect.com/science/article/pii/S0167739X12001318 doi: http://dx.doi.org/10.1016/j.future.2012.05.023

Fontaine, C., & Galand, F. (2007). A survey of homomorphic encryption for nonspecialists. *EURASIP Journal on Information Security*, *2007*. doi: 10.1155/2007/13801

Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., ... Stoica, I. (2009). Above the clouds: A Berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Tech. Rep. UCB/EECS*, *28*.

Gantz, J., & Reinsel, D. (2010). *The digital universe decade-are you ready (white paper)* (Tech. Rep.).

Gens, F. (2010). *Enterprise IT in the Cloud Computing Era* (Tech. Rep.). Retrieved from http://www.inst-informatica.pt/servicos/informacao-e-documentacao/dossiers-tematicos/teste-dossier-tematico-no-7-cloud-computing/tendencias/idc-enterprise-it-in-the-cloud-computing-era-new

Gohring, N. (2008). *Amazon's S3 down for several hours* (No. 28th Jan 2014). http://www.pcworld.com/businesscenter/article/142549/amazons_s3_down_for_severalhours. html. Retrieved from http://status.aws.amazon.com/s3-20080720.html

Goldreich, O. (1997). A sample of samplers - a computational perspective on sampling (survey). *Electronic Colloquium on Computational Complexity (ECCC)*, *4*(20), 24.

Goldreich, O., & Krawczyk, H. (1996). On the Composition of Zero-Knowledge Proof Systems. *SIAM Journal on Computing*, *25*(1), 169–192. Retrieved from http://epubs.siam.org/doi/abs/10.1137/S0097539791220688 doi: doi:10.1137/S0097539791220688

Goldreich, O., Micali, S., & Wigderson, A. (1991). Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM*, *38*(3), 690–728. doi: 10.1145/116825.116852

Goldreich, O., & Ostrovsky, R. (1996). Software protection and simulation on oblivious RAMs. *Journal of the Acm*, *43*(3), 431–473. Retrieved from <GotoISI>://WOS:A1996VA95700002 doi: Doi10.1145/233551.233553

Goldwasser, S., & Micali, S. (1982). *Probabilistic encryption & how to play mental poker keeping secret all partial information.* San Francisco, California, USA: ACM. doi: 10.1145/800070.802212

Gonçalves, V., & Ballon, P. (2011). Adding value to the network: Mobile operators' experiments with Software-as-a-Service and Platform-as-a-Service models. *Telematics and Informatics*, *28*(1), 12–21. Retrieved from http://www.sciencedirect.com/science/article/pii/S0736585310000365 doi: http://dx.doi.org/10.1016/j.tele.2010.05.005

Goodrich, M. T., Mitzenmacher, M., Ohrimenko, O., & Tamassia, R. (2012). *Privacy-preserving group data access via stateless oblivious RAM simulation.* Kyoto, Japan: SIAM.

Goodrich, M. T., Tamassia, R., & Schwerin, A. (2001). *Implementation of an authenticated dictionary with skip lists and commutative hashing* (Vol. 2). Anaheim, CA: IEEE. doi: 10.1109/DISCEX.2001.932160

Google docs, sheets, and slides size limits (Vol. 2015) (Web Page No. 07/01/2015). (2015, 2015). Google. Retrieved from https://support.google.com/drive/answer/37603?hl=en

Halevi, S., Harnik, D., Pinkas, B., & Shulman-Peleg, A. (2011). *Proofs of ownership in remote storage systems*. Chicago, Illinois, USA: ACM. doi: 10.1145/2046707.2046765

Hanser, C., & Slamanig, D. (2013). Efficient Simultaneous Privately and Publicly Verifiable Robust Provable Data Possession from Elliptic Curves. *IACR Cryptology ePrint Archive*, 392–406. Retrieved from http://eprint.iacr.org/2013/392

Harnik, D., Pinkas, B., & Shulman-Peleg, A. (2010). Side Channels in Cloud Services: Deduplication in Cloud Storage. *IEEE Security & Privacy*, *8*(6), 40–47. doi: 10.1109/MSP.2010.187

Höfer, C. N., & Karagiannis, G. (2011). Cloud computing services: taxonomy and comparison. *Journal of Internet Services and Applications*, *2*(2), 81–94. Retrieved from http://dx.doi.org/10.1007/s13174-011-0027-x doi: 10.1007/s13174-011-0027-x

Huang, Q., Yang, G., Wong, D., & Susilo, W. (2011). Efficient strong designated verifier signature schemes without random oracle or with non-delegatability. *International Journal of Information Security*, *10*(6), 373–385. Retrieved from http://dx.doi.org/10.1007/s10207-011-0146-1 doi: 10.1007/s10207-011-0146-1

Icart, T. (2009). *How to Hash into Elliptic Curves* (Vol. 5677). Santa Barbara, CA, USA: Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-03356-8_18 doi: 10.1007/978-3-642-03356-8_18

Jing, S.-Y., Ali, S., She, K., & Zhong, Y. (2013). State-of-the-art research study for green cloud computing. *The Journal of Supercomputing*, *65*(1), 445–468. Retrieved from http://dx.doi.org/10.1007/s11227-011-0722-1 doi: 10.1007/s11227-011-0722-1

Juels, A., & Kaliski Jr., B. (2007). Pors: Proofs of retrievability for large files. In *Proceedings of the acm conference on computer and communications security* (pp. 584–597).

Kamara, S., & Lauter, K. (2010). *Cryptographic Cloud Storage* (Vol. 6054). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-14992-4_13 doi: 10.1007/978-3-642-14992-4_13

Kate, A., Zaverucha, G., & Goldberg, I. (2010). Constant-Size Commitments to Polynomials and Their Applications. In M. Abe (Ed.), *Advances in cryptology - asiacrypt 2010* (Vol. 6477, pp. 177–194). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-17373-8_11 doi: 10.1007/978-3-642-17373-8_11

Khan, A. N., Kiah, M. L. M., Khan, S. U., Madani, S. A., & ur Rehman Khan, A. (2013). A study of incremental cryptography for security schemes in mobile cloud comput-

ing environments. In *Ieee symposium on wireless technology and applications* (pp. 62–67). doi: 10.1109/ISWTA.2013.6688818

Khan, A. N., Mat Kiah, M. L., Khan, S. U., & Madani, S. A. (2013). Towards secure mobile cloud computing: A survey. *Future Generation computer systems*, *29*(5), 1278–1299. Retrieved from http://www.sciencedirect.com/science/article/pii/S0167739X12001598 doi: http://dx.doi.org/10.1016/j.future.2012.08.003

Kristensen, M. D. (2007). Enabling cyber foraging for mobile devices. In *Proceedings of the 5th minema workshop: Middleware for network eccentric and mobile applications* (pp. 32–36). Citeseer.

Lin, S., & Costello, D. J. (2004). *Error Control Coding, Second Edition* (Vol. 123). Prentice-hall Englewood Cliffs, NJ.

Litwin, W., & Schwarz, T. (2004). Algebraic signatures for scalable distributed data structures. In *20th international conference on data engineering* (pp. 412–423). IEEE. doi: 10.1109/ICDE.2004.1320015

Liu, C., Chen, J., Yang, L., Zhang, X., Yang, C., Ranjan, R., & Rao, K. (2014, Sept). Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates. *Parallel and Distributed Systems, IEEE Transactions on*, *25*(9), 2234-2244. doi: 10.1109/TPDS.2013.191

Mandagere, N., Zhou, P., Smith, M. A., & Uttamchandani, S. (2008). *Demystifying data deduplication.* Leuven, Belgium: ACM. doi: 10.1145/1462735.1462739

Maniatis, P., Roussopoulos, M., Giuli, T. J., Rosenthal, D. S. H., & Baker, M. (2005). The LOCKSS peer-to-peer digital preservation system. *ACM Trans. Comput. Syst.*, *23*(1), 2–50. doi: 10.1145/1047915.1047917

Manvi, S. S., & Krishna Shyam, G. (2013). Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey. *Journal of Network and Computer Applications*, *41*(1), 424–440. Retrieved from http://www.sciencedirect.com/science/article/pii/S1084804513002099 doi: http://dx.doi.org/10.1016/j.jnca.2013.10.004

Marinos, A., & Briscoe, G. (2009). Community Cloud Computing. In M. Jaatun, G. Zhao, & C. Rong (Eds.), *Cloud computing* (Vol. 5931, pp. 472–484). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-10665-1_43 doi: 10.1007/978-3-642-10665-1_43

Marques, L., & Costa, C. J. (2011). *Secure deduplication on mobile devices.* Lisboa, Portugal: ACM. Retrieved from http://doi.acm.org/10.1145/2016716.2016721 doi: 10.1145/2016716.2016721

Mell, P., & Grance, T. (2011). The NIST definition of cloud computing (draft). *NIST special publication*, *800*, 145.

Merkle, R. C. (1980). Protocols for public key cryptosystems. In *Ieee symposium on security and privacy* (pp. 122–133). doi: 10.1109/SP.1980.10006

Meyer, D. T., & Bolosky, W. J. (2012). A study of practical deduplication. *Trans. Storage*, *7*(4), 1–20. doi: 10.1145/2078861.2078864

Miller, R. (2010). *Amazon Addresses EC2 Power Outages* (Vol. 2013) (No. 25th Sep 2013). http://www.datacenterknowledge.com/archives/2010/05/10/amazon-addresses-ec2-power-outages/. Retrieved from http://www.datacenterknowledge.com/archives/2010/05/10/amazon-addresses-ec2-power-outages/

Naone, E. (2010). *What Twitter Learns from All Those Tweets* (Vol. 2014) (No. 28th Jan 2014). MIT Technology Review. Retrieved from http://www.technologyreview.com/view/420968/what-twitter-learns-from-all-those-tweets/

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., & Zagorodnov, D. (2009). The Eucalyptus Open-Source Cloud-Computing System. In *9th ieee/acm international symposium on cluster computing and the grid* (pp. 124–131). Shanghai. doi: 10.1109/CCGRID.2009.93

Oprea, A., Reiter, M. K., & Yang, K. (2005). *Space-efficient block storage integrity*.

Paillier, P. (1999). Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *International conference on the theory and application of cryptographic techniques (eurocrypt '99)* (Vol. 1592, pp. 223–238). Prague, Czech Republic: Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/3-540-48910-X_16 doi: 10.1007/3-540-48910-X_16

Papamanthou, C., & Tamassia, R. (2007). Time and Space Efficient Algorithms for Two-Party Authenticated Data Structures. In S. Qing, H. Imai, & G. Wang (Eds.), *Information and communications security* (Vol. 4861, pp. 1–15). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-540-77048-0_1 doi: 10.1007/978-3-540-77048-0_1

Perez, J. C. (2014). *OneDrive's file size limit upped to 10GB, syncing speeds tripled* (Vol. 2015) (No. 07/01/2015). PCWorld. Retrieved from http://www.pcworld.com/article/2605912/onedrive-now-allows-files-of-up-to-10gb.html

Plank, J. S., & Ding, Y. (2005). Note: Correction to the 1997 tutorial on Reed-Solomon coding. *Software: Practice and Experience*, *35*(2), 189–194. doi: 10.1002/spe.631

Plank, J. S., & Xu, L. (2006). *Optimizing Cauchy Reed-Solomon Codes for Fault-Tolerant Network Storage Applications*. Cambridge, MA: IEEE. doi: 10.1109/nca.2006.43

Pugh, W. (1990). *Skip lists: a probabilistic alternative to balanced trees* (Vol. 33) (No. 6). doi: 10.1145/78973.78977

Reed, I. S., & Solomon, G. (1960). Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, *8*(2), 300–304. doi: 10.1137/0108018

Rivest, R. L., Adleman, L., & Dertouzos, M. L. (1978). On data banks and privacy homomorphisms. *Foundations of secure computation*, *32*(4), 169–178.

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, *21*(2), 120–126. doi: 10.1145/359340.359342

Sahai, A., & Waters, B. (2005). Fuzzy Identity-Based Encryption. In R. Cramer (Ed.), *Advances in cryptology – eurocrypt 2005* (Vol. 3494, pp. 457–473). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/11426639_27 doi: 10.1007/11426639_27

Sander, T., & Tschudin, C. (1998). Protecting Mobile Agents Against Malicious Hosts. In G. Vigna (Ed.), *International conference on mobile agents and security* (Vol. 1419, pp. 44–60). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/3-540-68671-1_4 doi: 10.1007/3-540-68671-1_4

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996). Role-based access control models. *Computer*, *29*(2), 38–47.

Savas, X, E., Ko, X00E, & K., C. (2010). Finite field arithmetic for cryptography. *IEEE Circuits and Systems Magazine, 10*(2), 40–56. doi: 10.1109/MCAS.2010.936785

Schwartz, M. J. (2012). *6 worst data breaches of 2011* (No. 20th Jan 2014). http://www.informationweek.com/news/security/attacks/232301079. Retrieved from http://www.informationweek.com/news/security/attacks/232301079

Schwarz, T. S. J., & Miller, E. L. (2006). Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage. In *26th ieee international conference on distributed computing systems* (p. 12). doi: 10.1109/ICDCS.2006.80

Setty, S., Blumberg, A. J., & Walfish, M. (2011). *Toward practical and unconditional verification of remote computations.* Napa, California: USENIX Association.

Shacham, H., & Waters, B. (2008). Compact Proofs of Retrievability. In *Advances in cryptology (asiacrypt)* (Vol. 5350, pp. 90–107). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-540-89255-7_7 doi: 10.1007/978-3-540-89255-7_7

Sheng-Cheng, Y., Wu-Hsiao, H., Ming-Yang, S., & Chun-Yuen, L. (2012). A study on the data privacy and operation performance for cloud collaborative editing systems. In *4th international conference on cloud computing technology and science (cloudcom)* (pp. 591–596). Taipei: IEEE. doi: 10.1109/CloudCom.2012.6427609

Shin, Y., Hur, J., & Kim, K. (2012). Security weakness in the Proof of Storage with Deduplication. *IACR Cryptology ePrint Archive*, 554. Retrieved from http://eprint.iacr.org

Singh, A., & Liu, L. (2008, April). Sharoes: A Data Sharing Platform for Outsourced Enterprise Storage Environments. In *2008 ieee 24th international conference on data engineering* (pp. 993–1002). Cancun: IEEE. doi: 10.1109/ICDE.2008.4497508

Sood, S. K. (2012). A combined approach to ensure data security in cloud

computing. *Journal of Network and Computer Applications*, *35*(6), 1831–1838. Retrieved from http://www.sciencedirect.com/science/article/pii/S1084804512001592 doi: http://dx.doi.org/10.1016/j.jnca.2012.07.007

Sookhak, M., Akhunzada, A., Gani, A., Khurram Khan, M., & Anuar, N. B. (2014). Towards Dynamic Remote Data Auditing in Computational Clouds. *The Scientific World Journal*, *2014*, 12. Retrieved from http://dx.doi.org/10.1155/2014/269357 doi: 10.1155/2014/269357

Sookhak, M., Gani, A., Talebian, H., Khan, S. U., Buyya, R., & Zomaya, A. Y. (2015). Remote Data Auditing in Cloud Computing Environments: A Survey, Taxonomy, and Open Issues. *ACM Comput. Surv.*, 36.

Sookhak, M., Talebian, H., Ahmed, E., Gani, A., & Khan, M. K. (2014, August). A review on remote data auditing in single cloud server: Taxonomy and open issues. *Journal of Network and Computer Applications*, *43*, 121–141. doi: 10.1016/j.jnca.2014.04.011

Storer, M. W., Greenan, K., Long, D. D. E., & Miller, E. L. (2008). *Secure data deduplication.* Alexandria, Virginia, USA: ACM. doi: 10.1145/1456469.1456471

Storm, D. (2011). *Epsilon breach: hack of the century?* (No. 28th Jan 2014). http://blogs.computerworld.com/18079/epsilon_breach_hack_of_the_century.. Retrieved from http://blogs.computerworld.com/18079/epsilon_breach_hack_of_the_century.

Subashini, S., & Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications*, *34*(1), 1–11. Retrieved from http://www.sciencedirect.com/science/article/pii/S1084804510001281 doi: http://dx.doi.org/10.1016/j.jnca.2010.07.006

Takabi, H., Joshi, J. B. D., & Gail-Joon, A. (2010). Security and Privacy Challenges in Cloud Computing Environments. *IEEE Security & Privacy*, *8*(6), 24–31. doi: 10.1109/MSP.2010.186

Vaquero, L., Rodero-Merino, L., & Morán, D. (2011). Locking the sky: a survey on IaaS cloud security. *Computing*, *91*(1), 93–118. Retrieved from http://dx.doi.org/10.1007/s00607-010-0140-x doi: 10.1007/s00607-010-0140-x

Wang, C., Chow, S., Wang, Q., Ren, K., & Lou, W. (2012). Privacy-Preserving Public Auditing for Secure Cloud Storage in Cloud Computing. *IEEE Transactions on Computers*, *PP*(99), 1–14. doi: 10.1109/TC.2011.245

Wang, C., Wang, Q., Ren, K., Cao, N., & Lou, W. (2012). Toward secure and dependable storage services in cloud computing. *IEEE Transactions on Services Computing*,, *5*(2), 220–232. doi: 10.1109/tsc.2011.24

Wang, C., Wang, Q., Ren, K., & Lou, W. J. (2009). Ensuring Data Storage Security in Cloud Computing. *Iwqos: 2009 Ieee 17th International Workshop on Quality of Service*, 37–45. Retrieved from <GotoISI>://000274551300005

Wang, H. (2012). Proxy Provable Data Possession in Public Clouds. *IEEE Transactions on Services Computing,, PP*(99), 1. doi: 10.1109/TSC.2012.35

Wang, Q., Wang, C., Li, J., Ren, K., & Lou, W. J. (2009). Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing. *Proceedings of Computer Security (Esorics 2009)*, *5789*, 355–370. Retrieved from <GotoISI>://000274011000022

Wang, Q. A., Wang, C., Ren, K., Lou, W. J., & Li, J. (2011). Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing. *Ieee Transactions on Parallel and Distributed Systems*, *22*(5), 847–859. Retrieved from <GotoISI>://WOS:000288755400012 doi: 10.1109/Tpds.2010.183

Wei, L., Zhu, H., Cao, Z., Dong, X., Jia, W., Chen, Y., & Vasilakos, A. V. (2013). Security and privacy for storage and computation in cloud computing. *Information Sciences*(0). Retrieved from http://www.sciencedirect.com/science/article/pii/S0020025513003320 doi: http://dx.doi.org/10.1016/j.ins.2013.04.028

Whaiduzzaman, M., Sookhak, M., Gani, A., & Buyya, R. (2014). A survey on vehicular cloud computing. *Journal of Network and Computer Applications*, *40*, 325–344. Retrieved from http://www.sciencedirect.com/science/article/pii/S1084804513001793 doi: http://dx.doi.org/10.1016/j.jnca.2013.08.004

What's the maximum file size i can upload? (Vol. 2015) (Web Page No. 07/01/2015). (2014, 2014). Retrieved from https://support.box.com/hc/en-us/articles/200520238-What-s-the-maximum-file-size-I-can-upload-

Whittaker, Z. (2012). *Amazon web services suffers partial outage.* (No. 28th Jan 2014). http://www.zdnet.com/blog/btl/amazon-web-services-suffers-partial-outage/79981: ZDNet.

Xie, M., Wang, H., Yin, J., & Meng, X. (2007). Integrity auditing of outsourced data. In *Proceedings of the 33rd international conference on very large data bases* (pp. 782–793). Vienna, Austria: VLDB Endowment.

Yan, Z., Hongxin, H., Gail-Joon, A., & Mengyang, Y. (2012). Cooperative Provable Data Possession for Integrity Verification in Multicloud Storage. *IEEE Transactions on Parallel and Distributed Systems,, 23*(12), 2231–2244. doi: 10.1109/tpds.2012.66

Yang, K., & Jia, X. (2012). Data storage auditing service in cloud computing: challenges, methods and opportunities. *World Wide Web*, *15*(4), 409–428. Retrieved from http://dx.doi.org/10.1007/s11280-011-0138-0 doi: 10.1007/s11280-011-0138-0

Yang, K., & Jia, X. (2013). An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing. *Ieee Transactions on Parallel and Distributed Systems*, *24*(9), 1717–1726. doi: 10.1109/TPDS.2012.278

Yeh, S.-C., Su, M.-Y., Chen, H.-H., & Lin, C.-Y. (2013). An efficient and secure approach for a cloud collaborative editing. *Journal of Network and Computer Applications*, *36*(6), 1632–1641. Retrieved from http://www.sciencedirect.com/

science/article/pii/S1084804513001409 doi: http://dx.doi.org/10.1016/j.jnca.2013.05.012

Yuan, J., & Yu, S. (2013a). *Proofs of retrievability with public verifiability and constant communication cost in cloud.* Hangzhou, China: ACM. doi: 10.1145/2484402.2484408

Yuan, J., & Yu, S. (2013b). Secure and Constant Cost Public Cloud Storage Auditing with Deduplication. *IACR Cryptology ePrint Archive, 2013*, 149.

Zhang, J., & Mao, J. (2008). A novel ID-based designated verifier signature scheme. *Information Sciences*, *178*(3), 766–773. Retrieved from http://www.sciencedirect.com/science/article/pii/S0020025507003581 doi: http://dx.doi.org/10.1016/j.ins.2007.07.005

Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, *1*(1), 7–18. Retrieved from http://dx.doi.org/10.1007/s13174-010-0007-6 doi: 10.1007/s13174-010-0007-6

Zhang, Y., & Blanton, M. (2012). Efficient Dynamic Provable Possession of Remote Data via Update Trees. *IACR Cryptology ePrint Archive*, 291. Retrieved from http://eprint.iacr.org/

Zheng, Q., & Xu, S. (2011, February). Fair and dynamic proofs of retrievability. In *Proceedings of the first acm conference on data and application security and privacy - codaspy '11* (p. 237). New York, New York, USA: ACM Press. Retrieved from http://dl.acm.org/citation.cfm?id=1943513.1943546 doi: 10.1145/1943513.1943546

Zheng, Q., & Xu, S. (2012). *Secure and efficient proof of storage with deduplication.* San Antonio, Texas, USA: ACM. doi: 10.1145/2133601.2133603

Zhibin, Z., & Dijiang, H. (2012). Efficient and secure data storage operations for mobile cloud computing. In *8th international conference and workshop on systems virtualiztion management network and service management* (pp. 37–45).

Zhifeng, X., & Yang, X. (2013). Security and Privacy in Cloud Computing. *IEEE Communications Surveys & Tutorials*, *15*(2), 843–859. doi: 10.1109/SURV.2012.060912.00182

Zhu, Y., Wang, H., Hu, Z., Ahn, G.-J., & Hu, H. (2011). Zero-knowledge proofs of retrievability. *Science China Information Sciences*, *54*(8), 1608–1617. Retrieved from http://dx.doi.org/10.1007/s11432-011-4293-9 doi: 10.1007/s11432-011-4293-9

Zissis, D., & Lekkas, D. (2012). Addressing cloud computing security issues. *Future Generation computer systems*, *28*(3), 583–592. Retrieved from http://www.sciencedirect.com/science/article/pii/S0167739X10002554 doi: http://dx.doi.org/10.1016/j.future.2010.12.006